APRIL/MAY 1984 An independent magazine published by ECC Publications

new

SINCLAIR PROJECTS

95p

THE COMPLETE HARDWARE COMPANION

ONE MAN AND HIS QL — MEET DAVID KARLIN

EXCLUSIVE OFFER ZX-81 KIT FOR ONLY £25

QL REVIEW
ROBOT ARM
FLEXIBLE FILING
PRECISE MATHS
CENTRONICS INTERFACE
NEW: HARDWARE REVIEWS

The most imaginative graphics programming software for your Spectrum

PAINTBOX is a must for every 48K SPECTRUM owner! If you want to exploit the full graphics capability of your machine you can do so . . . simply and easily with PAINTBOX.

Take a look at the screen prints shown above. They are the sort of thing you could produce on your SPECTRUM. PAINTBOX gives you an entire suite of graphics programming aids in one integrated program. For instance:

UDG EDITOR

The facility to define and re-define up to 84 graphics characters for your programs or for use in screen-planning.

UDG DRAWING BOARD

A fully-integrated UDG planner for defining up to 4 Banks of characters. Planning facilities include MIRROR, INVERSE, ROTATE, FILE.

SKETCHPAD

An experimentation "window" for developing the UDG set.

PRECISION PLOTTER

An amazingly versatile high resolution drawing board which includes PAPER choice INK choice PLOT, DRAW, CIRCLE, FILL, ARC, OVER, ERASE, and STORE!

SCREEN PLANNER

Combining PRECISION PLOTTER and UDG characters! A multi-purpose graphics facility to enable you to produce incredible screen graphics. All work can be sent to a Printer and SAVED as SCREEN or as CODE with its own built-in machine code routine for instant recall from BASIC.

The program comes with a DEMO program and a 28 page book that's packed with hints and tips on how to get the best from PAINTBOX.

PAINTBOX is ideally suited for use with the Print 'n' Plotter Spectrum Jotter Package — the first and best graphics planning pad for the Spectrum! So why not place an order today? Write, call, or see your local dealer.

01-660 7231

24 HOUR CREDIT CARD ORDERING

Post to: Dept S Print 'n' Plotter Products Ltd., 19 Borough High Street, London SE1 9SE.
Please send me:

- .... "PAINTBOX" SOFTWARE @ £7.50 (+75p p+p total £8.25)
- .... SPECTRUM JOTTER PADS @ £7.50 (£1.50 p+p total £9)
- .... SPECTRUM KEYBOARD OVERLAYS @ £2.60 (35p p+p total £2.95)
- .... 5 ROLLS ZX PRINTER PAPER @ £11.55 (95p p+p total £12.50)

☐ I enclose remittance in full
☐ Please bill my Access/Barclaycard/Visa/Mastercard No:

____

Overseas orders please add 25% for additional surface mail rate.

NAME ____________________

ADDRESS ____________________

____

DEALERS: Phone 01-403 6644 for enquiries

Print 'n' Plotter Products

SINCLAIR PROJECTS

YOUR *Sinclair Projects* has changed this month. To reflect the interest and needs of readers, we have expanded coverage of matters in the Sinclair hardware market and made improvements to the traditional areas of the magazine.

Our aim is to make *Sinclair Projects* the complete hardware companion for your Sinclair machine — essential reading for those who want to know how commercial hardware works and how you can build something yourself.

To achieve it we will be reviewing in depth all the latest peripherals on the Sinclair market, showing what they do, of what they comprise, and how they perform their tasks. This month our chief reviewer, John Lambert, looks inside the Dean Electronics thermal printer and the Fox Electronics programmable joystick for the Spectrum, among many other items.

We also intend to provide lengthy reviews of important products or comparisons of groups — of peripherals which perform similar tasks. In this issue Mike Wright considers the possibilities of the new QL, the Sinclair Quantum Leap.

The world of computers can often seem an inhuman place but without people there would be no machines, no programs, no users. From now we will be promoting the human aspect by meeting some of the people behind the latest machines or developments.

Everyone is talking about the QL at the moment, so Nigel Clark went to speak to one of the men behind its development. Though still a young man, David Karlin has already done a great deal in the world of micro-electronics.

The number of build-it-yourself projects has been reduced from six to four but they are still as interesting as in previous issues. With the growing interest in robots, as shown by the number of readers who have enquired about the Prowler, we have a project to build a simple arm. We have made the device as uncomplicated as possible, so that we give the essentials of what is involved while still making something useful.

In a project for the Spectrum, Richard Sargent shows how to build a one-chip Centronics interface. That is the industry standard interface for printers and should interest anyone wanting to improve hard copies of information.

The other two articles are software projects. Both have been chosen because of their help to serious users of the Spectrum. One permits people to build a flexible filing system which can be used to accommodate a variety of records. The other shows how accurate mathematics can be performed.

Managing editor Nigel Clark **Consultant editor** David Buckley **Managing production editor** Harold Mayes MBE **News writer** John Lambert **Design** Elaine Bishop **Advertisement manager** John Ross **Advertisement Executive** Robert Marcus **Editorial assistant** Colette McDermott **Production assistant** Dezi Epaminondou **Managing director** Terry Cartwright **Chairman** Richard Hease.

Sinclair Projects is published bi-monthly by ECC Publications Ltd. It is in no way connected with Sinclair Research Ltd.

Telephone, all departments: 01-359 3525. If you would like to contribute to any of the *Sinclair User* group of publications please send programs, articles or ideas for hardware projects to Sinclair User and Projects, ECC Publications, 196-200 Balls Pond Road, London N1 4AQ. We pay £50 per 1,000 words for each article used.

ISSN 0264/0449. Printed and typeset by Cradley Print PLC, Warley, W. Midlands. Distributed by Spotlight Magazine Distribution Ltd, 1 Benwell Road, Holloway, London N7. 01-607 6411.

Power from the ZX-81

DESPITE having large amounts of computing power at their disposal, many large companies are using the ZX-81 to help them with their research. They have discovered that the little machine, with the addition of the Forth ROM cartridge from Skywave Software, gives them many facilities at a low cost — the ROM costs only £25.

David Husband of Skywave, based at Boscombe, Bournemouth, says he was surprised at first when he started receiving orders from large companies such as ICI, universities and a number of Government establishments. "It does not surprise me any more; they just realised what can be done with the ZX-81 and the ROM," he says.

The ROM with its multi-tasking capability makes the ZX-81 a useful controller of applications, particularly in laboratories and as a teaching aid. Husband says it would be ideal for a project such as the weather station being built in *Sinclair Projects*, where the measurements of temperature and pressure could be performed at regular intervals but the user could still program the machine. It is also possible to have a number of windows on the screen in the same way as the new QL.

QL orders rush

SIR CLIVE SINCLAIR'S reputation for launching good products is such that orders for the QL have been flooding in. The only machines to have been seen so far are those on display at the launch yet there is now a waiting list "well into the thousands".

Although the first machines were expected to be delivered early in March, the unexpected rush means that a backlog will develop.

There were more than 400 orders in the first two days after the launch and now they are arriving at 500 a day.

ZX-Microdrive design frozen

THE DESIGN of the Spectrum Microdrive has been frozen and there are no plans to make the improvements which have been made for the QL Microdrive. A spokesman for Sinclair Research said most of the changes had been cosmetic but admitted that they allowed an additional 15K of storage to be guaranteed for the QL version.

The ZX Microdrive sold for the Spectrum has a minimum of 85K of storage, whereas the QL Microdrive could supply 100K.

David Karlin, one of the leading members of the QL design team, said that a number of engineering changes had been made and there had been a great deal more error-checking and attempts to increase the performance of the controller chip.

A spokesman added that the changes would be difficult to implement on the ZX Microdrive.

Plea for rampack

AS A fairly new owner of a ZX-81, I am still using the unexpanded 1K version. Now, however, I wish to expand to the much more useful 16K package. Nowhere can I find a circuit diagram for an add-on RAM pack. Can you advise me of anyone who can supply a RAM pack in kit form? The object, obviously, is to do the job as cheaply as possible.

R J F Richardson,
Harrogate.

● *A 16K RAM pack is a complicated circuit and since they can be obtained ready-built from as little as about £15 it is not worth considering trying to make one. The parts alone would probably cost as much.*

More support call

HAVING bought the October/November issue for the purpose of constructing the Battery-Backed RAM, the first three-quarters of a page were completely over my head. I was therefore very disappointed to find that not only was I unable to understand the article but the RAM board was of a rather strange capacity i.e., 8KB.

When programming my ZX-81, using the 16K programs freely available in the computer magazines, I frequently experience problems with supply voltage fluctuations and computer screen lock-up. The only way to restore computer functions is to re-set the mains supply, therefore losing the program, if not already lost.

I understand that the problems are usually found with the ZX-81 but you can see that this problem requires a 16K battery-backed RAM. Is it possible to modify the 8K RAM project to function as a 16K board by using more 2K RAMs or different components?

K J Bryan,
Deal, Kent.

● *You misunderstand the purpose of the battery-backed RAM. The project was to allow one to store machine code routines rather than Basic programs and is mapped outside the Basic program area. To overcome your problem, you could build the battery-back-up system of the December/January issue.*

Teleprinter printer

LOOKING through *Sinclair Projects* and considering building the Radio Teleprinter, I decided to write a short machine code subroutine to simulate the output from the interface. It showed two problems with page 17, the machine program which I have not noticed in Update:

Line 16514 Data : missing a 00. Dec. address 16523 should read 16528.

At 16600 the call address given may be satisfactory for old ROM users but it caused me a few headaches. I think that new ROM users should have :

16600 40D8 CD 2B 0F CALL SLOW.

Malcolm Purves,
Bristol.

Cheap connector for the Spectrum

THE NEW I/O port from Multitron gives the Spectrum a means of communicating with the outside world. Using the port it could control motors, turn lights on and off, or detect when a switch has been closed. What it does is to transfer signals to and from the outside in a form the CPU can understand.

The port is an uncased PCB with through connector based on the Intel 8255 AP-5 chip. The chip has three 8-bit ports — A, B and C — and a control register D, the addresses being 31, 63, 95 and 127 respectively. Each port can be set to either input or output, with the upper and lower nibbles of port C capable of being set independently to either.

Two more modes of operation are available which allow strobed I/O with handshaking and strobed bi-directional operation; in both cases the data can be latched. Details are given in the user manual.

Connections to the board are either by a 28-way Spectrum-style edge connector or Soldercon pins — breadboard style. The manual gives comprehensive details of how the port works and how to set it up. It also includes two brief programs, one to make the port test itself and one to show binary numbers being output to LEDs.

One thing it does not do is to give simple circuit diagrams to show how to connect a LED or perhaps a relay.

Priced at £13.50 plus 35 pence p&p, including manual, it provides a cheap introduction to control applications. Available from Multitron, 5 Milton Close, Headless Cross, Redditch, Worcs B97 5BQ. Tel: 0527 44785.

Sinclair thermals

DEAN ELECTRONICS has recently introduced a Sinclair-compatible thermal printer, the Alphcom 32. It is manufactured by the American company which produces the Timex-Sinclair 2040.

The printer plugs into the rear connector of either the ZX-81 or Spectrum and will accept the standard commands of LPRINT, LLIST and COPY, so existing software can be used without alteration. Using 110mm. wide white thermal paper, it produces a very readable output at a speed of roughly two lines per second.

The printer casing is approximately 195mm. × 140mm. × 55mm. black-moulded ABS, with a perspex blister on top which holds the paper. Two wires emerge from the back, one — about 150mm. long — to an over-size edge connector containing a 74LS10, used to decode A7 and A2, and a ferrite ring to suppress interference. It has a ZX-81 size connector to the computer and a through port for RAM packs. The other lead connects to the supplied external power supply by way of a male 3.5mm. jack plug. Inside the printer there is a minimum of electronics — a ROM chip, marked TS2040, to handle the printer operations; five chips to control the printing mechanism; a handful of discrete components; and two PCB-mounted switches to turn the printer on and off and to advance the paper. Use of both switches together performs a self-test function.

The bulk of the space is occupied by a very solid-looking, rubber-mounted printer mechanism. The printhead is made of a ceramic material into which 20 wires are inlaid. As they are moved across the paper they burn off the top surface of the paper to leave a black ink impression.

Each wire covers two character squares in a zig-zag fashion which shows up the only disadvantages. When doing a COPY the zig-zag is noticable on any solid blocks of ink.

The printer becomes warm in use but that is not a problem, as there are adequate ventilation slots on the top and bottom and a large heatsink inside. On a Spectrum the edge connector lead fouls the power lead, making insertion difficult and it does not fit flush at the bottom, making the Spectrum slightly unstable.

Costing £59.95, including power supply and one roll of paper, with extra rolls of paper at only £1, the printer must be seen as a good alternative to the Sinclair printer. The Alphacom 32 is obtainable from Dean Electronics Ltd, Glendale Park, Fernbank Road, Ascot, Berkshire SL5 8JB. Tel: 0334 885661 and branches of W H Smith.

Sound made in stereo

FOR THOSE with a musical bent who have a Sinclair machine, help is at hand. Not a musical bent straightener but the Tricord from Petron is a stereo programable sound generator board in two versions, with and without an internal amplifier and speaker, for both the ZX-81 and Spectrum.

The Trichord has three basic modes of operation. First, using an in-built PROM, it can reproduce any of 255 sound effects ranging from one described as a low bong to a steam engine and whistle, plus many indescribable ones. Second, it can be used to play three-part harmony and, finally, the internal registers of the PSG chip can be accessed to produce your own sound effects.

All versions of the Trichord are the same-sized black plastic box which has a ZX-81 connector and through port. On a Spectrum that means that only a Sinclair printer could be plugged into the back of it.

Inside the box is an AY-3-8910 PSG chip. It has 14 internal registers to control the frequency and pitch of three sound channels, the pitch and channel of a white noise generator, separate volume controls, and has eight in-built envelope shapes for which the period can be altered.

The Trichord is probably the most versatile sound generator on the market at the price.

Petron Electronics is at Courtlands Road, Newton Abbot, Devon TQ12 2JA. Tel: 0626 62836.

Disc driving on the Spectrum

LATEST in a sudden crop of disc interfaces for the Spectrum is the FDC-1 Mk2 from Technology Research. It will accept up to two 5¼in. drives in either 40- or 80-track, single- or double-sided format and is complete with a utility disc.

The interface plugs on to the rear user port and provides a through port for other add-ons, drive cable and connector and a socket for the Spectrum power supply. On power-up the contents of an EPROM in the interface is loaded into the upper 4K of memory and a jump is made to the DOS, where a password has to be entered. The password has to agree with a password held on the disc to allow access.

At that point the full range of commands becomes available — LOAD/SAVE of Basic or machine code, both of which can be auto-run; ERAse a file on disc; and an INITialisation routine for new discs. The initialisation is carried-out after a new disc has been FORMATed using a program on the utility disc and stores the current password on the disc.

Initialising a disc will wipe it, so an additional command LOCK is provided to prevent that happening, if required. DIRectory will give you — provided you have the correct password — a list of the files and their length, plus the amount of spare space. Two additional commands are available for random access of the disc, PUT and GET, but our provisional copy of the new instructions gave limited details, although we are assured that will be remedied on the proper instructions.

Inside the interface are two PCBs; the lower one takes the lines across to the rear connector and holds the power socket. The upper one holds the main electronics, a disc operating chip — a 1771 — the EPROM and a good deal of buffering around the cable socket. The 1771 is a relatively old disc chip and cannot provide double density but by using two boards it is very easy to change one when, at a later stage, you need that feature. The buffering on the cable is particularly useful, as the interface uses the same standard as BBC machines and a drive can be disconnected without crashing the system.

In use, the interface proved reliable. The only time it crashed was when trying to save a program without giving it details of program length and start. In that instance the interface defaults to saving the whole 64K but the drive did not appear to like the idea and just spun aimlessly. Apart from that it worked first time, every time.

Machine code users who normally use the upper memory for their routines are catered for as Technology Research can, for a nominal fee, provide the DOS assembled anywhere in the upper 32K.

With a 40-track, single-sided drive the interface gives 97.5K of file space, with 2.5K being taken by the directory. Larger-capacity drives lose a similar percentage.

Priced at £85 plus VAT, the interface is rather expensive but it allows the use of drives not dedicated to one machine; also if an 80-track, double-sided drive is used, you have 390K of file space at less than £2 a time.

Technology Research Ltd, 356 Westmount Road, London SE9 1NW. Tel: 01-856 8408.

Joystick variety

NEW FROM Fox Electronics is a programmable joystick interface for the Spectrum. The interface plugs into the rear connector of the Spectrum and has a through connector for other add-ons. On the right-hand side of the case is a standard 9-pin, D-type, Atari-style socket for the joystick and one switch.

To use the interface all you have to do is put up the switch, which then displays a menu on the screen. You then have the option of creating a new key set from any of the 40 keys, including the shift keys and ENTER, or selecting, with a single keystroke, one of the 16 sets already created.

Pressing the E key exits to Basic ready to load a game and programs the joystick. If necessary, the key sets can be saved on tape.

Leaving the switch down will make the Spectrum ignore the interface unless you are using another add-on which uses the ROMCS line; if so, you may find that a clash occurs.

Inside the interface is a 2K CMOS RAM, the 6116LP, and a small ni-cad battery. When in use the battery is kept topped-up by the Spectrum power supply, via a 7805 regulator, and on power-down maintains the memory for a minimum of six months.

On putting up the switch the interface pages-out the Spectrum ROM and jumps to the program held in its RAM. The program then transfers into the Spectrum RAM, pages the ROM back in and puts the menu on the screen. On pressing the E key, the program transfers back to its own RAM, sets up the joystick and calls the NEW routine. Any new key sets created are saved in the process.

All that is clever and provides the easiest interface on the market today but it also provides two by-products. First, when the switch is put down, the interface causes a hardware re-set. That is to say if you have a game running you can jump out of it without having to pull the plug — a saving of plug wear. Second, details are available from Fox for a machine code programmer to use the interface as a pseudo ROM. Often-used routines could then be loaded at the flick of a switch.

When used with a Microdrive, the Spectrum power lead fouls the joystick lead, making insertion difficult. At £28.50, the interface is very good value. From Fox Electronics Ltd, 141 Abbey Road, Basingstoke, Hampshire RG21 9ED. Tel: 0256 20671.

REVIEWS

Kempston interface prints defined graphics

NEW from Kempston Electronic is the model E Centronics interface for the Spectrum. It contains an EPROM which enables it, on power-up, to direct the commands LPRINT and LLIST direct to the printer without the need for additional software. Also built into the EPROM are routines which allow the use of COPY for the Epson and Siekosha range of printers.

Housed in the standard Kempston case, the interface plugs into the user port of the Spectrum and is complete with a cable to connect to the printer.

The interface is dead-ended in that it does not have a through port for other add-ons. That may be a problem if you want to use it at the same time as the Kempston joystick interface or any other dead-ended device. A difficulty which may arise with full-size keyboards is that the case is shaped with a lip to fit on top of the standard Spectrum. The lip may prevent the case fitting snugly.

Inside the case is a 2K EPROM which houses the printer software and a handful of chips which detect when the Spectrum is using the LPRINT, LLIST and COPY commands. It does that, for example, with the COPY command, by monitoring an address, and when the Spectrum uses it to do a COPY the interface takes over and directs output to the printer.

Using the interface is very simple. If only text is to be printed, no setting-up is needed, apart from POKEing an address with the number of columns required if that is other than the default setting of 80. The address is one of the unused ones in the system variables area.

To use the COPY command the interface must be set up for your type of printer. By entering as a direct command COPY: REM? the interface displays a menu page which shows its current status. You can then set it up for the Epson range, Seikosha 100 or 250 printers, or add your own routines. Once you have set it up in that way any user-defined characters or graphics characters which appear in a listing will be printed as shown on the screen An annoying feature is that those characters are wider than normal and make the listing appear untidy.

For computer artists there is an enlarged setting. When it is turned on, COPY will produce a double-sized copy, about 180mm. × 145mm. on an Epson, suitable for hanging on the wall. Other settings are available to control the tokens, escape characters and automatic line feeds.

At £55 inc. the interface is by no means inexpensive but has many useful features. Details from Kempston Micro Electronics Ltd, Unit 30, Singer Way, Woburn Road Industrial Estate, Kempston, Bedford MK42 7AF. Tel: 0234 856633.

QL

Real computing power at a moderate cost

Mike Wright reviews the latest machine from Sinclair Research and finds much of which Sir Clive can be proud.

THE NEW MACHINE from Sinclair Research, the QL or Quantum Leap, promises to live up to its name and to be a major revolution for people in business who want real computing power at moderate cost.

The hardware is designed merely to provide large computing power at a reasonable cost but the supporting software and expansion facilities are all aimed at the professional and small business market.

INTEL 8049

The QL measures 5⅜in. × 1¾in. × 18¾in., weighs a little more that 3lb., and can be connected to either a monitor or a television screen. A colour monitor will give a wider screen and a greater resolution than a television screen. The machine features the Motorola 68008 32-bit processor with 128K of user RAM; 32K is reserved for the screen display, which in its highest resolution gives 512×256 pixels in four colours or 26×256 pixels and eight colours.

An Intel 8049 is also used in the CPU controlling the keyboard, sound, RS232C receive and real-time clock functions. That leaves the 68008 to look after all the prinicipal functions and in all there is 1MB of non-segmented address space available.

Four custom-built chips are also included. The first, dual-sourced from Plessey and Synertek, controls display and memory; the second, dual-sourced by NCR and Synertek, controls the other major functions, including the Microdrives, local area network and RS232C transmission; the third and fourth from Ferranti provide analogue functions required by the Microdrives.

There is a 65-key, full-travel keyboard. The introduction of the keyboard means that the traditional use of keywords on other Sinclair machines will not be possible. It is complete with two built-in Microdrives and the ability to connect six more. They are improved versions of Spectrum Microdrives and are not compatible with the Spectrum, although with re-formatting the cartridges will be.

MICRODRIVES

Other features include a ROM cartridge slot which will allow the ROM to be expanded by 32K, an expansion slot for a 0.5MB RAM, two RS232C ports and two joystick ports. The QL should also be able to link to 63 other QLs or Spectrums.

The cost of all those facilities is a reasonable £399 but before it can be put to use a monitor or a television set and a printer will be needed. That will add about £300 for a colour monitor and £250 for a reasonably good-quality dot matrix printer.

Even with those additions, the cost compares very favourably to existing systems. Sir Clive Sinclair said at the launch: "For £800 you can have a word processor better than anything you can buy currently."

Despite those cost-benefits there are several points which must be worrying to potential business users. The biggest of them is the decision to stay with the Microdrive, although in an updated and improved form, instead of using floppy discs as back-up storage. Since the introduction of Microdrives last year, some experts have been worried by the performance of the drives and by the use of continuous loop of video tape.

In addition, there is no connection for a cassette recorder. Although loading from a cassette is a slow, tedious business, a cassette copy of a program is usually fairly reliable and it is cheaper to produce commercial programs on cassette.

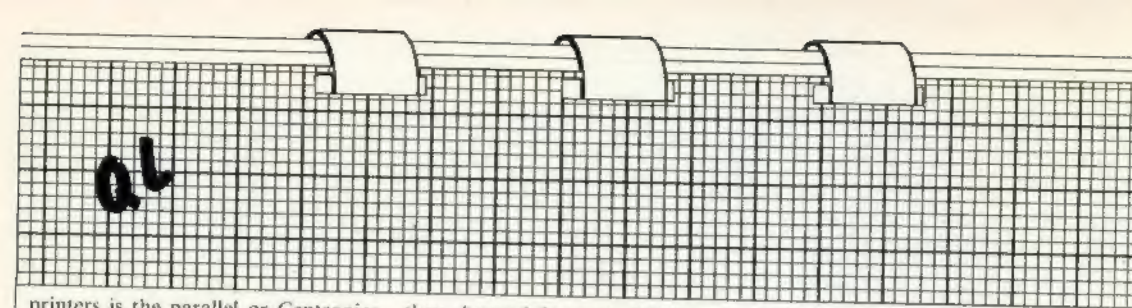
The cost of a blank cassette can be as little as 15 pence if bought in large numbers, while the costs of a Microdrive cartridge is about £5.

DISC PLAN

It is also interesting to note that Sinclair Research plans to produce a hard disc interface, while it has no plans to produce either a disc drive or a floppy disc interface. The cost of a hard disc could be two to three times the cost of the computer.

Another point to consider for users with large amounts of data is that once the RAM expansion has been fitted the QL has 640K of RAM. Each Microdrive cartridge holds only a maximum of 100K.

In keeping down the costs of the new machine, Sinclair may have sinned by omission for business users. The industry standard interface for



printers is the parallel or Centronics interface.

The QL is equipped with two RS232C ports but not a Centronics port. Most printers are fitted with a Centronics interface while the RS232C is offered as a more expensive option. An alternative would be to wait until Sinclair produces its planned Centronics interface but that again means extra cost.

For its operating systems, Sinclair Research has developed QDOS for which there are some elaborate claims. It is able to run more than one program at a time, it can divide the screen into a number of windows so that different displays can be shown simultaneously and input-output is device-independent.

The language used is another in the growing family of Basics, this one being called Sinclair Super Basic. It is said to be "a radical enhancement of Spectrum Basic." That makes for the same problems people found when they changed from the ZX-81 to the Spectrum. They will not be able to upgrade their machines while retaining their favourite software. Even if the program is on a Microdrive cartridge, it will not be possible to use it on the QL.

The feature which will interest non-technical users is the suite of programs written by Psion specifically for the QL. They are described by Psion managing director David Potter as "more powerful and functional than existing products for desk-top computers costing up to £5,000."

OPTIONS DISPLAYED

The suite has been designed for usability by a mass market with no prior training. It is said that even the most inexperienced person can perform useful tasks immediately, while experienced users can achieve a remarkable level of sophistication.

There is a word processor, a spreadsheet, a database and a business graphics program. They are integrated in style, structure, design and, perhaps most important, in the sharing of information. The last feature allows data to be transferred between programs so that information from the database or spreadsheet can be transferred to the graphics program where it can be represented graphically and from where it can be moved into a document for printing.

Although the manual contains large sections on all four programs, information on the present status and options available are displayed in English at the top of the screen.

The QL appears to live up to Sir Clive's claims that it is a quantum leap for the company and computing. It is aimed at the business market and it would appear to satisfy the demands of people in business. Sinclair Research, however, appears to be hedging its bets by including joystick ports, so that games can be played on it.

PROFILE

Counting the pins helps in making the quantum leap

THE MOST DIFFICULT job in designing the new Sinclair QL was to reduce the pin count to a satisfactory level. That is the view of one of the leading members of the design team, David Karlin.

"We spent the first two months trying to reduce the pin count to 80," he says, adding that it was one of the major reasons for choosing the Motorola 68008 for the main CPU chip. Using the ability of that chip to process information in 32 bits but having only an 8-bit bus allowed for a big reduction in the number of pins.

32/8-BIT

The chip, however, caused Sinclair Research some problems in deciding how to describe the machine. Ideally it should be a 32/8-bit machine but that is not a generally-accepted naming system and it was thought it would have been confusing.

Karlin adds that whatever it was called, all the software would look like that for a 32-bit machine.

The full CPU consists of the 68008 operating at 7.5MHz for all the principal functions, while a second processor, the Intel 8049, controls the keyboard, sound, RS232C receive and real-time clock functions.

The operating system, called QDOS and developed by Sinclair, is said to include a number of key features such as single-user multiple tasking, time-sliced priority job scheduler, display handling for multiple-screen windows and device-independent input-output.

It has 1MB of non-segmented address space, which makes possible a wide family of peripherals and enhancements. Of that, 32K is used for the screen bit map; a small amount is used for other functions leaving, on the unexpanded 128K machine, about 96K of usable memory.

The RAM can be extended externally to 640K and the 32K ROM can be expanded by ROM cartridge to 64K. The QL uses Super Basic which is said to be a great improvement on the Basic used in the Spectrum.

There are four other chips which are designed to Sinclair specifications. Two have been dual-sourced to ensure there are no difficulties with delivery. The first, which controls the display and memory, is supplied by Plessey and Sunertek. The other, from NCR and Synertek, controls the other major functions, including the two Microdrives, local area network

and RS232C transmission. The other two chips are supplied by Ferranti and provide the analogue functions required by the Microdrives.

Karlin says that once Sinclair had set the specifications for the chips discussions were started with a number of manufacturers to discover not only if they could meet the technical requirements but also if they would be able to supply the chips in sufficient numbers. Having decided on the companies, the design work was done independently.

"Rather than have one company do the design work and then supply the other with a mask, we thought it better to have each company do its own design," he says.

That was an area which threatened to become one of major difficulty. When the first prototypes were received from the manufacturers they did not work in the system. "We checked them thoroughly and eventually found that there was nothing wrong with them. The problems had been external," Karlin says.

He followed the development work all the way through and listening to him talking about it would be easy to think that he had been involved in nothing more than a making a few simple improvements to the Spectrum. Many observers, however, are already saying that Sinclair Research is fully justified in saying that the new machine is a quantum leap of similar proportions to that which brought computing power within the reach of millions with the introduction of the ZX-80.

Existing machines which use similar technology cost a minimum of £3,500 and to upgrade existing micros to provide comparable facilities it is estimated the cost would be almost £2,000. The QL with its software support costs £399.

GENERAL RESEARCH

Karlin has been with Sinclair Research since August, 1982 when he returned from the States, where he had been working on general research for Fairchild in Palo Alto, California. Although born and educated in Britain, he went to the U.S. after leaving Trinity College, Cambridge, where he had been studying a mix of engineering and electrical sciences.

He decided he wanted to return to Britain, contacted Sinclair Research which was looking for researchers at the time, and was given a job. Since then he has worked on a number of projects but his main work has been on the development of the QL.

From conception to launch took only 14 months. "The first six

PROFILE

months were spent designing the ICs and the last eight months were needed to debug them," he says.

In that time, little changed from his original ideas of what the computer should contain. "I would have liked to include a £20 colour tube and we decided to increase the RAM to 128K at a fairly late stage but the overall design was much the same as my original ideas," he says.

He emphasises, though, that the design was a co-operative effort by a number of people at Sinclair Research.

PORTABLE MACHINE

"It was not a simple matter of Sir Clive giving us a specification and the rest of us producing a machine to satisfy that," he says. "We were all throwing around ideas, some of which we used and others we did not."

One of the early ideas, given wide publicity by Sir Clive, was a portable machine with its own power supply and flat-screen television and two Microdrives. Karlin said, however, that that had been decided against early in the development stage.

Having two Microdrives formed part of the final machine, although improvements have been made on those which are used with the Spectrum. Although much of the drives has remained the same, with storage in the region of 100K and an average access time of 3.5 seconds, Karlin says they have been "improved a great deal".

"We made a number of engineering changes, put in a great deal more error-checking and tried to increase the performance of the controller chip."

He cannot say if those improvements will also be made to the ZX Microdrives.

Karlin defends the decisions to omit two facilities, a Centronics interface and a cassette recorder connector. With the QL having a ROM cartridge slot and Microdrives, he sees no necessity for inputting information from cassette.

The reasons for including an RS232C rather than a Centronics interface were more complicated. While agreeing that the Centronics is more usual at the moment, he says:

"The Centronics interface is more expensive and occupies more board area than the RS232C but does exactly the same job. The problem is that it is the industry standard but we think that with our using the RS232C more people will start using it as well."

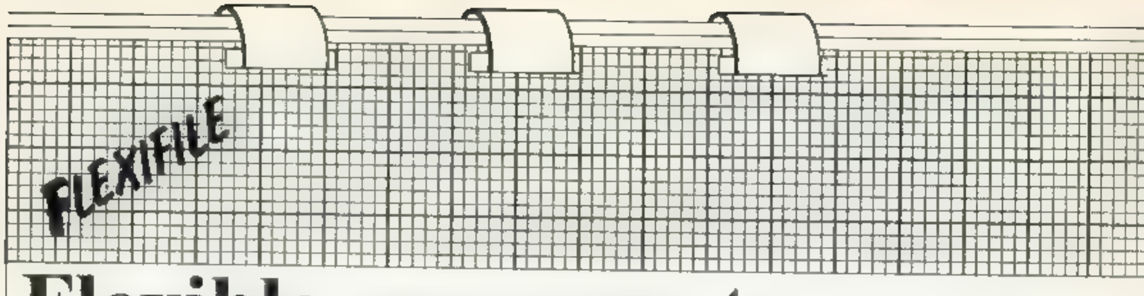
Karlin is confident that the QL represents 14 months well-spent. He sees an immediate market for the machine in the professions and higher education, particularly for university students who have large calculations to do or theses to write. In addition, there is the small business market where he thinks large numbers can be sold, not only because of the price but "because of the large amount of business software and the quality of that software."

PLENTY OF IDEAS

Sinclair Research is also confident about the machine and it intends to publish ROM information as soon as it can get it together from all the internal documents.

"Everything we have done has been done very carefully so that we can be confident it will work," Karlin says. "We did not publish that information about the other machines because we wanted to be able to change things if we found it necessary."

About any future work with which he is involved at Sinclair Research, Karlin is keeping very quiet. "There are always plenty of ideas being thrown around at Sinclair, so I do not think there will be any shortage of work," he says.



Flexible response to growing piles of paper

Filing large amounts of information can be difficult with many of the available programs. John Davison decided to write his own to allow him to adjust the system to suit his requirements

THROUGH THE YEARS I seem to have amassed a considerable quantity of paperwork containing information on a wide variety of topics, most of which I would not want to dispose of, but to which I seldom refer. While I was anxious to keep the information, I was equally keen to reduce the volume of paper. The obvious solution was to store it on cassette tape.

In looking for a suitable file program I soon realised that most of those available, or published, had one of two failings of a pre-determined fixed length and/or a large number of relatively short records. In both respects that was the precise opposite of what was called for; I needed to store fairly long individual records and have total flexibility of record length. The answer, of course, was to write the program to suit my requirements. **Flexi-File** is the result.

TEXT FILES

Initially I used it to create files of historical and geographical information but given its inherent flexibility I have since used it for a variety of other types of information, including temporary text files such as the first draft of an article. There is no restriction as to its use. If you have any information in the form of a subject heading — record name — and textual details record entry, Flexi-File is for you.

In addition to reducing the sheer bulk of your filing system, Flexi-File offers basic text-handling functions. Let me make it clear now that the program is not a word processor; to achieve that on the Spectrum requires a machine code program. While I admire several features exceptional to Sinclair Basic, not least the string-handling utilised to the full in this program, its speed is not impressive when compared, for instance, to BBC Basic. Ironically though the Spectrum offers much more user-available RAM than its competitors and it is that feature that makes it so useful for file-handling, permitting a file approaching 30K in length.

OVER-WRITE

Flexi-File offers a record displayed in a formatted style, to avoid splitting words at line ends, and with the option for a printout from a ZX printer if available; an editing facility which enables text to be inserted, deleted, or moved, and individual characters to be over-written.

I have found the program especially useful when writing notes or original text. Instead of the confusion of erasing and alterations one associates with a 'paper file' it has been simple to change the text as needed and then print-out a clean copy via the printer. It is obviously an ideal program for anyone making notes for educational purposes, be it a school project, university dissertation, or preparation for a lecture; the first draft of the various sections can be typed into the file and edited as required.

Clearly the ZX printer is no use to produce the final copy but at least the screen display offers a clean, tidy and readable draft from which the final version can be typed — no struggling with deciphering your own hastily-scribbled handwriting. Or are we, perhaps, already approaching the day when written work will be accepted on computer cassette?

The only function usually associated with file programs and not available here is a SORT routine but, given that Flexi-File is designed to hold a relatively small number of records, that is not a significant disadvantage. If we assume an average record length of 200 words — 1,200 bytes — the file will hold 25 records and it will take the search routine about one second to locate the start of the last record in the file, which should be acceptable.

So let us look at the program in action, which should enable you to see what personal applications you have for Flexi-File. Having typed-in the program listing, carefully and correctly, enter the following as a direct command:

LET e$="FLEXI-FILE": LET fe=30000: SAVE e$ LINE 80: VERIFY e$ <enter>

Once the program — about 5.7K — is SAVEd and VERIFYed, enter GO TO 80 and the menu will appear; the program is operated primarily from the menu. The BORDER and PAPER are set to blue and the INK to white, chosen because they seem to be the most pleasing to read. They can be changed easily by altering the appropriate numbers in line 80 but bear in mind that there are other colour commands in the program which assume white INK, so beware of producing invisible prompts.

FACILITIES

The facilities offered are to LOAD a file from cassette, SAVE the current file to cassette, list the record names in the current file, change one of those record names, start a new record, or review an existing record. The bottom line of the display invites you to type-in a number, 1 to 6, or 'n' to create a new file. So type 'n' and the screen will clear and print a warning message, appropriately in red.

That is just a safeguard in case you inadvertently press 'n' while an existing file is in the computer. Type 'c'

FLEXIFILE

The Listing

```
  10 PAUSE 0: IF INKEY$="" THEN GO TO 10
  11 LET i$=INKEY$: BEEP .002,0: RETURN
  40 CLS : GO SUB 600: PRINT PAPER 2;AT 0,CODE f$(n);" : ";Page;TAB 27;30000-fe: LET c=LEN a$: IF c>672 THEN LET c=672
  41 PRINT a$( TO c)
  42 LET line=10: LET col=16
  43 PRINT AT line,col; INVERSE 1; OVER 1;" "
  44 GO SUB 10: PRINT AT line,col; INVERSE 1; OVER 1;" "
  45 IF i$>"4" AND i$<"9" THEN LET line=line+(i$="6")-(i$="7"): LET col=col+(i$="8")-(i$="5")
  46 LET line=line-(21 AND line>21)+(21 AND line<1): LET col=col-(32 AND col>31)+(32 AND col<0)
  47 IF (line-1)*32+col+1>LEN a$ THEN LET line=INT (LEN a$/32)+1: LET col=LEN a$-(line-1)*32
  48 LET Posn=(line-1)*32+col+1
  49 IF CODE i$=226 THEN RETURN
  50 IF i$="d" THEN INPUT AT 1,0;"How many characters? ";ch: LET a$=a$( TO Posn-1)+a$(Posn+ch TO ): GO TO 40
  51 IF i$="s" THEN LET start=Posn: GO TO 42
  52 IF i$="f" THEN LET finish=Posn: LET x$=a$(start TO finish): LET a$=a$( TO start-1)+a$(finish+1 TO ): GO TO 40
  53 IF i$="m" THEN GO TO 57
  54 IF i$="r" THEN PRINT AT line,col;" ";CHR$ 8;: BEEP .1,20: GO SUB 10: PRINT i$: LET a$(Posn)=i$: GO TO 42
  55 IF i$<>"i" THEN GO TO 43
  56 INPUT AT 0,0;"Enter new text : "; LINE v$
  57 LET a$=a$( TO Posn-1)+v$+a$(Posn TO ): GO TO 40
  80 BORDER 1: PAPER 1: INK 7: GO SUB 450: PRINT e$;AT 0,16;30000-fe;" bytes free"''"1 - LOAD a File"''"2 - SAVE the File"''"3 - List Record Names"''"4 - Change a Record Name"''"5 - Start new Record"''"6 - Review a Record";AT 21,0;"Type number : N = Open New File "
  81 GO SUB 10: IF i$="n" OR i$="N" THEN GO TO 90
  82 IF i$<"0" OR i$>"6" THEN GO TO 81
  83 GO TO 1000*VAL i$
  90 GO SUB 450: PRINT PAPER 7; INK 2;AT 10,4;"ANY EXISTING INFORMATION";AT 12,7;"WILL BE LOST WHEN";AT 14,5;"A NEW FILE IS CREATED";AT 21,0;" c=Continue : Other keys=Return ": GO SUB 10: IF i$<>"c" OR i$="C" THEN GO TO 80
  91 GO SUB 700: DIM f$(30000): LET fe=2: PRINT INVERSE 1;AT 10,6;"* FILE IS NOW OPEN *": PAUSE 30: BEEP .5,20: GO TO 80
 100 LET rlen=CODE f$(n-2)*256+CODE f$(n-1): RETURN
 150 LET f$(n-2)=CHR$ (INT (rlen/256)): LET f$(n-1)=CHR$ (rlen-CODE f$(n-2)*256): RETURN
 200 INPUT AT 0,0;"Enter Record Name"'">"; LINE n$: IF LEN n$>24 THEN CLS : GO TO 200
 205 PRINT AT 12,0;"Is this the correct name?"'' PAPER 2;n$
 210 GO SUB 10: IF i$<>"y" AND i$<>"Y" THEN GO TO 200
 215 LET n$=CHR$ LEN n$+n$: RETURN
 250 LET a=n+CODE f$(n)+1: RETURN
 300 LET a$="": BEEP .5,20
 305 GO SUB 10: IF CODE i$=226 THEN RETURN
 310 IF CODE i$=12 THEN PRINT CHR$ 8;" ";CHR$ 8;: LET a$=a$( TO LEN a$-1)
 315 IF CODE i$>31 AND CODE i$<128 THEN PRINT i$;: LET a$=a$+i$
 320 IF LEN a$=672 THEN BEEP .1,10: BEEP .1,20: BEEP .1,30: BEEP .1,20: BEEP .1,10: RETURN
 325 GO TO 305
 350 LET rlen=rlen+diff: GO SUB 150: LET fe=fe+diff: RETURN
 400 INPUT AT 0,0;"Enter Record Name"' LINE z$: LET n=5
 405 IF n>fe THEN GO TO 420
 410 IF f$(n+1 TO n+CODE f$(n))=z$ THEN RETURN
 415 GO SUB 100: LET n=n+rlen: GO TO 405
 420 CLS : PRINT AT 6,0;"There isn't a Record Name"'"spelt like > "'' INVERSE 1;"""";z$;"""";, INVERSE 0: PRINT ,AT 12,0;"Would you like a list of the    Record Names? ";AT 21,0;"y = YES : other keys = NO": GO SUB 10: IF i$="y" OR i$="Y" THEN GO TO 3000
 425 GO TO 80
 450 BRIGHT 0: CLS : BRIGHT 1: RETURN
 500 IF diff>0 THEN GO TO 530
 510 LET b1=start
 515 LET b2=b1+1000: IF b2>fe THEN LET b2=fe
 520 GO SUB 550: LET b1=b2+1: IF b1>fe THEN RETURN
 525 GO TO 515
 530 LET b2=fe
 535 LET b1=b2-1000: IF b1<start THEN LET b1=start
 540 GO SUB 550: LET b2=b1-1: IF b2<start THEN RETURN
 545 GO TO 535
 550 LET t$=f$(b1 TO b2): LET f$(b1+diff TO b2+diff)=t$: RETURN
 600 CLS : PRINT PAPER 2;f$(n+1 TO n+CODE f$(n)): LET line=1: RETURN
 700 CLS : INPUT "Name of FILE? "; LINE e$: IF LEN e$>10 THEN GO TO 700
 710 RETURN
1000 GO SUB 700: PRINT AT 10,12-INT (LEN e$/2);"LOADing ";e$; INVERSE 1;AT 15,9;"START THE TAPE": LOAD e$ DATA f$(): LET fe=256*CODE f$(1)+CODE f$(2): GO TO 80
2000 CLS : LET f$(1)=CHR$ INT (fe/256): LET f$(2)=CHR$ (fe-CODE f$(1)*256): PRINT INVERSE 1;AT 6,6;"SAVEing ";e$: SAVE e$ DATA f$(): BEEP .5,20
2010 GO SUB 450: PRINT AT 6,8;" REWIND  TAPE  ";AT 9,4;"Press any key to VERIFY": PAUSE 0: GO SUB 450: PRINT AT 10,1;"On Error ""R"" enter GO TO 2000": VERIFY e$ DATA f$(): BEEP .1,10: BEEP .15,20: BEEP .15,30: GO TO 80
3000 LET n=5: LET a=0
3010 GO SUB 450: PRINT PAPER 2;e$;" : Record Names                "(1 TO 32-LEN e$): LET line=2
3020 LET a=a+(a=0)-(a=1): PRINT PAPER a;AT line,0;f$(n+1 TO n+CODE f$(n)): GO SUB 100: LET n=n+rlen
3030 IF n>fe THEN GO SUB 3100: GO TO 80
3040 LET line=line+1: IF line=22 THEN GO SUB 3100: LET line=2: GO TO 3010
3050 GO TO 3020
3100 INPUT AT 1,0;"Print-out wanted? "; LINE i$: IF i$="y" OR i$="Y" THEN COPY
3110 RETURN
4000 GO SUB 450: GO SUB 400: PRINT AT 2,0;"Current name is :-"; PAPER 0''z$: GO SUB 200: GO SUB 100: LET diff=(LEN n$-1)-LEN z$: IF diff=0 THEN LET f$(n TO n+CODE n$(1))=n$: GO TO 80
4010 LET start=n+CODE f$(n)+1: GO SUB 500: LET f$(n TO n+CODE n$(1))=n$: GO SUB 350: GO TO 80
5000 CLS : GO SUB 200: CLS : PRINT PAPER 2;n$(2 TO );TAB 27;30000-fe: GO SUB 300: LET rlen=3+CODE n$(1)+LEN a$: LET n=fe+3: GO SUB 150: LET f$(fe+3 TO fe+rlen)=n$+a$: LET fe=fe+rlen: GO TO 80
6000 CLS : PRINT INVERSE 1;AT 0,9;"REVIEW ROUTINE": GO SUB 400: GO SUB 100: PRINT AT 12,15-INT (LEN z$/2);">";z$;"<"
6010 LET Page=1: GO SUB 250: INPUT AT 0,0;" Edit : Format : Add : Delete : Return  > "; LINE i$: IF i$="a" OR i$="A" THEN GO TO 8000
6020 IF i$="d" OR i$="D" THEN GO TO 9000
6030 IF i$="e" OR i$="E" THEN GO TO 7000
6040 IF i$="f" OR i$="F" THEN GO TO 6100
6050 IF i$="r" OR i$="R" THEN GO TO 80
6060 GO TO 6010
6100 GO SUB 600: PRINT PAPER 2;TAB 30;Page;","
6110 LET b=a+31: IF b>n+rlen-3 THEN LET b=n+rlen-3
6120 IF f$(b)<>" " AND f$(b)<>"-" AND f$(b+1)<>" " AND b<>n-3+rlen THEN LET b=b-1: GO TO 6120
6130 PRINT AT line,0;f$(a TO b)
6140 LET a=b+1: IF a>n+rlen-3 THEN GO SUB 6200: GO TO 6010
6150 IF f$(a)=" " THEN LET a=a+1: GO TO 6150
6160 LET line=line+1: IF line<22 THEN GO TO 6110
6170 GO SUB 6200: LET Page=Page+1: GO TO 6100
6200 INPUT AT 1,0;"COPY : LPRINT > "; LINE i$: IF i$="c" OR i$="C" THEN COPY
```

FLEXIFILE

```
6310 IF i$="l" OR i$="L" THEN LPRINT "-";f$(n+1 TO n+CODE f$(n));"-";f$(n+CODE f$(n)+1 TO n+rlen-3)
6320 RETURN
7000 LET b=a+671: IF b>n+rlen-3 THEN LET b=n+rlen-3
7010 LET a$=f$(a TO b): GO SUB 400
7020 LET diff=LEN a$-(b-a+1): IF diff=0 THEN GO TO 7040
7030 LET start=b+1: GO SUB 500
7040 LET f$(a TO a+LEN a$-1)=a$: GO SUB 350: LET a=a+LEN a$: IF a>n+rlen-3 THEN GO SUB 250: GO TO 6010
7050 LET page=page+1: GO TO 7000
8000 CLS : GO SUB 600: PRINT PAPER 2;TAB 27;30000-fe: GO SUB 300: LET start=n+rlen-2: LET diff=LEN a$: GO SUB 500: LET f$(start TO start+LEN a$-1)=a$: GO SUB 350: GO SUB 250: GO TO 6010
9000 GO SUB 450: PRINT AT 10,15-INT (LEN z$/2);">";f$(n+1 TO n+CODE f$(n));"<";AT 14,6; INVERSE 1;"Delete this Record ?": GO SUB 10
9010 IF i$<>"y" AND i$<>"Y" THEN GO TO 6010
9020 LET diff=rlen-rlen*2: LET start=n+rlen-2: GO SUB 500: LET fe=fe+diff: GO TO 80
```

and you are asked to INPUT the filename (e$); as that is used in the LOADing and SAVEing routines it cannot exceed 10 characters in length and the program refuses to accept an invalid file name — lines 700-710.

The file array (f$) is then DIMensioned 30,000 characters long and the file-end marker (fe) set at 2. That last variable is used to point to the end of that part of the file already used, or in other words fe + 1 is the first free byte. It is set initially at 2 because the first two bytes of a file hold the value of fe.

fe = CODE f$(1) ★ 256 + CODE f$(2).

Once again the screen is cleared and the menu printed. This time type '5' to enter the first record. In common with most of the routines in the program, this single-line routine — line 5000 — makes extensive use of sub-routines. The first of those — line 200 — INPUTs the record name, which cannot exceed 24 characters in length, for reasons connected with the display routines. Before the name, (n$), is accepted it is printed so that a check can be made of its spelling.

SPELLING

Most of the choices make use of the sub-routine at line 10, which obviates the need to use the ENTER key; a repeat facility is available in this sub-routine, on all keys, by holding down the key, but note that with a SHIFTed character the SHIFT key has to be kept held down also. Where a YES/NO response is expected, 'y' = YES and all other keys = NO.

So, if the record name is spelt correctly, type 'y' and the record name is accepted, with a control character added at the start of string — line 215 — the CODE of this character represents the length of the name and its relevance will be apparent later.

The record name is then printed at the top of the screen on a strip of red PAPER, along with the number of free bytes in the file. As the record name has not yet been placed in the file, its length is not included in fe and so the number displayed at this stage should be 29998. The sub-routine at line 300 is then called and you can type the entry on to the screen, the entry being held in a temporary string (a$). It is displayed on the screen exactly as it will be held in the file and individual characters may be deleted by using the DELETE key in the usual way.

TYPE THE ENTRY

A screenful of text is the most that may be entered by this routine and when the length of a$ reaches 672 — line 320 — the program BEEPs to inform you of the fact and returns to the main routine in line 5000. To return before a$ reaches that length type SYMBOL SHIFT/A (STOP) — line 305. Before the record is entered into the file its overall length is calculated (rlen), that being stored in the first two free bytes — line 150 — followed by the record name (n$) and the entry (a$). Thus a complete record comprises the following elements:

(CODE byte 1) ★ 256 + (CODE byte 2) = total record length

(CODE byte 3) = length of record name

(bytes 4 to x) = record name

(bytes x + 1 to y) = record entry

Finally this routine adds the record length to fe and returns to the menu.

To test the program it will be worth repeating the procedure a few times, so that all functions can be examined.

END MARKER

Having done that and having returned to the menu display, type '3' to list the record names. That routine prints a heading, on red PAPER, of the filename, which is presumed to be at least five characters long; if you anticipate shorter file names add extra space between the quotation marks in line 3010 and then print-out the record names alternately on blue and black PAPER.

If there are more than 20 records the listing is displayed as two pages with an option to send each page to the printer. This routine involves the search sub-routine.

Initially the variable n is set at 5 — line 3000 — which is the subscript of f$ holding the length of the first name. After that name has been printed, n is incremented by the length of the first record — lines 100 and 3030 — so that it equals the subscript holding the second name's length, and so on until it exceeds the file-end marker. Thus the time taken to locate any record depends on its position in the file; the time taken to find the 'last' record is a product of the number of records, irrespective of their individual or combined lengths.

ROUTINE

The routine at line 4000 — type '4' from the menu — may at first glance appear superfluous after all; the record name entry sub-routine gives you a chance to check that the name is correct. That routine is designed to permit a complete change of the name. I first included it as a result of storing biographical information about fictional characters; the record name was, of course, the name of the character and in a few instances I wanted to change the name. Depending on your personal applications, this routine may or may not be used much.

Given that the new record name will not necessarily be the same length as the old one, it becomes necessary to move blocks of the file up or down within f$ to create extra space or close vacated space. The sub-routine at line 500 performs that function and is entered with the variable diff equal to the difference in length of the two names, and start pointing to the first subscript after the old name. The file is moved in blocks of 1001 bytes and the new name inserted. The record length control characters and fe are adjusted accordingly — sub-routine 350 — and the menu re-displayed.

The principal routine of the program is that entered by typing '6' from the menu, starting at line 6000, the Review Routine. You are asked to INPUT the name of the record to be reviewed and when that has been located in the file, five options are displayed in the lower part of the screen — Edit, Format, Add, Delete and Return.

The choice is made by INPUTing, in either lower- or upper-case, the initial letter — which appears in INVERSE VIDEO, line 6010; at the risk of stating the obvious, when typing-in the program those letters are preceded by CAPS SHIFT/4 and followed by CAPS SHIFT/3. As you would expect 'r' is used to quit the routine and returns you to the menu.

'E' takes you into the editing facility — lines 7000-7050 — and begins by assigning appropriate values to a and b to create a temporary string, a$, comprising a maximum of 672 characters from the chosen record.

The display has the record name, 'page' number and number of free bytes as a heading, on red PAPER, followed by the text held in a$; the editing cursor is primed at the centre of the screen. The cursor effectively inverts the INK and PAPER colours in its current position and hence shows up clearly amid a screen of text.

The editing procedures are contained in a sub-routine starting at line 40, which was inspired by a useful routine published in *Sinclair Projects*, January, 1983. Strictly speaking, as the sub-routine is called only from line 7010 it should have been incorporated into the main routine but I included it in several programs and have become accustomed to it being 'GOSUB 40', so did not bother to renumber it for this particular use. The main point is that it does not detract from usefulness of the program.

FORGOTTEN

While in the sub-routine — lines 40 to 57 — in the main file is forgotten; all operations work on a$. The functions available are deleted, insert, move, and replace. The cursor is moved by using the cursor control keys — 5 to 8 — but UNshifted. Line 46 treats each line and column of the display as if it were a loop; so, for instance, moving the cursor down from the bottom line causes it to reappear on the topmost line of the same column, and vice versa.

That means that it is sometimes quicker to reach a particular character by apparently moving in the opposite direction. That is possibly as clear as mud, in which case I suggest that you play with the cursor and you will soon see the point I am making.

Having positioned the cursor, typing 'd' — delete, line 50 — produces a prompt in the lower screen asking you to INPUT the number of characters to be deleted, the first one being at the cursor position. Almost instantaneously, the screen is re-displayed with the selected characters removed from a$.

To move a block of text, begin by moving the cursor to the first character of the block and type 's' — start, line 51. Then move the cursor to the last character of the block and type 'f' — finish, line 52. As with the delete function, a$ is re-printed with the block of text removed. That block is stored as x$ and can be moved to any 'page' of any record in the file, or indeed in any other file.

There is one very important proviso, that neither the block-move nor the insert functions should be used again until the block has been re-inserted, otherwise it will be lost. From another viewpoint that can be useful, as move can be utilised effectively to delete the block of text — quicker than counting the characters. To re-locate the block position the cursor and type 'm' — line 53 — the block will be inserted immediately before the cursor-position and a$ reprinted.

To replace an individual character, position the cursor at the character and type 'r' — line 54; a space is printed over the character and a BEEP invites you to type-in the new one, which is printed into the space and inserted at the appropriate position in a$.

If the key pressed was 'i', a prompt appears in the lower screen asking you to type-in the new text insert — line 56 — and when ENTER is pressed that will be inserted immediately before the cursor position as with the block move.

COMPARED

When the editing of a particular 'page' is complete type STOP (SYMBOL SHIFT/A). The length of a$ is then compared to its original length line 7020 — and if it is different the sub-routine at line 500 is called to move the file up or down to accommodate the new version.

The record control characters and fe are increased or decreased accordingly. The variables a and b are then incremented and the next 'page' of the record displayed.

If during that routine you should delete some text by mistake, do not

Summary of main variables

c$:	the filename (< 11 characters long)
f$:	the file itself (up to 30000 characters)
fe:	the file-end marker (stored as f$(1 TO 2))
n$:	record name (< 25 characters long)
a$:	text being entered or edited
i$:	character returned from INKEY$ sub-routine (line 10)
t$:	a temporary string holding up to 1001 characters, used during the 'file-shift' sub-routine.

press STOP. So long as you are still within the sub-routine, only a$ is being affected; the file (f$) remains in its original form. Instead, press CAPS SHIFT/SPACE — BREAK — and then enter GO TO 7000 to start the editing function.

PAGE NUMBER

After the final 'page', the main review routine prompt appears in the lower screen. Having edited a record you will probably want to see it formatted, so type 'f'. The format function — lines 6100 to 6220 — also prints a heading on red PAPER, comprising the record name and a page number. Note that any given record may produce more pages in that form than in the editing style, because on average a formatted screen will contain only 620 characters from the file, as against a full editing display of 672.

Also note that the PRINT statement in line 6100 assumes that no record will produce more than nine pages; if you think that may not be the case, change that statement to read — PRINT PAPER 2; TAB 29+(page<10);

The record is then displayed one line at a time, each line being checked to ensure that no words carry-over from one line to the next. Each time the screen is filled there is the option to make a hard-copy on the ZX printer. Typing 'c' — COPY — does precisely that, printing-out a reproduction of the screen; 'l' — LPRINT — provides a complete printout of the record but that will be unformatted. When the whole record has been displayed the main prompt re-appears.

This time, enter 'a' and the single-line routine — 8000 — enables you to enter an addition to an existing record. This screen display is the same as when starting a new record and the new text is added to the end of the existing entry, for which reason the first character to be typed should almost certainly be a space, which will save you that portion of editing later.

To enter the text, type STOP (SYMBOL SHIFT/A); this has been used on a number of occasions where you might have expected ENTER to have been used. I decided against using that key because of its proximity to the space key, which increases the chance of catching it accidentally with potentially-irritating if not disastrous consequences.

Again, the sub-routine at line 500 is used to move the file down to make room for the extra text and the appropriate control characters and fe adjusted.

The last function to consider is selected by entering 'd' — Delete, line 9000+. That function displays the record name and asks if you wish to delete the record, a safeguard in case 'd' was entered in error. Assuming you do, the file is shifted up to overwrite the unwanted record and fe decreased by the record length.

If the record was the last one in the file, it just decreases the value of fe and the record will be over-written when a new record is entered. The function returns you direct to the main menu — line 80 — as no more operations are possible on a non-existent record.

It is probably worth noting the two escape routes, to avoid a state of panic should you feel that you have entered something incorrectly and are about to lose hours of careful typing of valuable information.

BREAK

Whenever the sub-routine at line 10 is being used, typing CAPS SHIFT/SPACE, (BREAK), will produce report code L — BREAK in program. This can be especially useful during the editing routines. When a letter or name is being INPUT typing CAPS SHIFT/6 — cursor down — produces report code H — STOP in INPUT — from which you can enter an appropriate GO TO statement or examine the file via direct commands.

When a file has been reviewed fully you will obviously want to SAVE it on to tape cassette. Most file programs use a separate array to store record names, which requires you to save the program with each file. That is the main advantage of Flexi-File. By incorporating record names, and the file-end marker, in the main file array it becomes possible to store files separate from the program, with a small saving in the time it takes to LOAD and SAVE any given file.

Typing '2' from the menu selects the single-line routine at line 2000 which, having SAVEd the file array, BEEPs and prints an appropriate message to prompt you to re-wind the tape to VERIFY the file. Immediately prior to SAVEing the file, the current value of fe is inserted into the first two bytes of f$.

There is obviously no need to SAVE the program. To LOAD a file back into the computer, type '1' from the menu display. The routine at line 1000 begins by asking you to input the file name and refuses to accept one longer than 10 characters; no file can have been SAVEd with a name longer than that. Once the file is LOADed, the appropriate value of fe is calculated from the first two bytes of the file and the menu re-displayed.

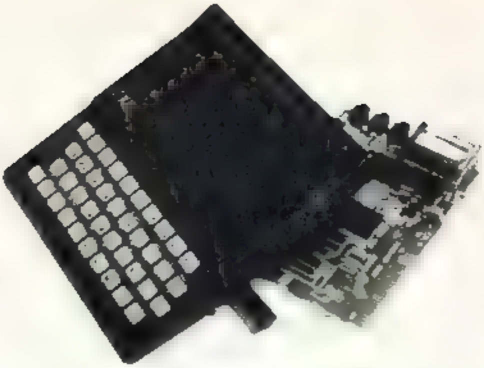
APPLICATION

Having tried all the routines on your test data, you should have a good idea of the flexibility and application of the program. Although written for, and so far used only on tape cassette, there is no reason why Flexi-File should not be used with the Microdrive.

Using the EDIT routine to move a block of text from one file to another is clearly a very slow process with cassette — possible but not very practical.

Given the speed of the Microdrive, however, it becomes a potentially very useful function, with up to three files on one cartridge. Of course, it will become possible to amend the program to take full advantage of the improved facilities offered by the Microdrive.

If, like myself to date, you have not received an order form for the Microdrive and Interface, you can at least still make a start on creating useful files with the benefit of full text-editing and know that they can be transferred to the mass-storage system when they are available.



Build your own Sinclair Special ZX-81 Kit Offer ONLY £25 (plus p&p)

A special offer open to readers of *Sinclair Projects* has been negotiated which means you can buy the world-beating ZX-81 for just £25 (plus post and packing).

Stocks are limited, so be sure to place your order soon. Allow 28 days for delivery. Maximum four units per applicant.

The kit is available by mail order only. No callers please. The prices apply to United Kingdom only. Overseas orders can be accepted but there will be an extra postal charge. The full price can be obtained on application to ECC Publications at the address on the coupon.

To: Sinclair Projects Special Offer,
ECC Publications, 196-200 Balls Pond Road, Islington, London N1 4AQ

Please send me ________ ZX-81 kit(s) at the special *Sinclair Projects* price of £25 plus £2.95 p&p.

Please tick if you require a VAT receipt ☐

*I enclose a cheque/postal order payable to ECC Ltd for £ ________

*Please charge to my Access/Barclaycard/Trustcard account no. ________

*Please delete/complete as applicable

Signature ________

Name Mr/Mrs/Miss ________

Address ________

AGF

MICRODRIVE

PROGRAMMABLE JOYSTICK INTERFACE

for

Spectrum

or ZX81

ONLY 26·95 +£1·00 p&p

AGF PROGRAMMABLE INTERFACE

Recognised as the only true Hardware Programmed joystick interface this product offers all the features associated with such a design.

You can use *any* Atari-compatible joystick controller with *any* software for your Sinclair Spectrum or ZX81, not just those with a joystick option.

Movement of the joystick is recognised by the computer *exactly* the same as pressing the appropriate control keys, and can therefore give the most immediate response to that movement. The hardware programmed design works with *all* possible key-reading methods, both BASIC and Machine Code.

Eight directional movement, with or without the fire button being pressed, can be achieved by only programming the left, right, up, down and fire keys required by the game.

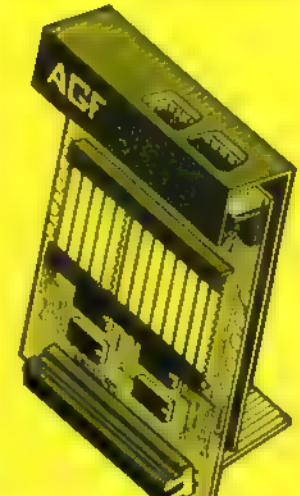
Programming is achieved by a two-digit code, which is looked up on the Programming Chart supplied, for each direction and firing button. These two numbers are then selected on a pair of leads which are clipped onto appropriately numbered strips on the interface.

Once configured this can be marked onto a Quick Reference Programming Card for storing with the game. As the programming is *not* power dependent the interface can be immediately used when next switched on.

The keyboard remains fully functional and can be used simultaneously with the joystick.

An integral rear expansion connector means there is no need to remove the interface to connect other peripherals.

NB. A recent design improvement now means that the AGF Programmable Interface works with the new Quickshot II rapid "Auto Fire" feature.



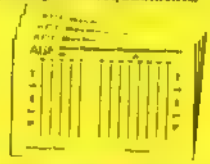
KEY FEATURES

- Programmable design gives TOTAL software support.
- Accepts Atari, Competition Pro, Wico, Starfighter, Quick Shot, Le Stick etc.
- Rear extension connector for all other add-ons.
- Free demo program and instructions.

PACKAGE CONTENTS SUPPLIED

- Programmable Interface Module as illustrated, complete with clip-on programming leads.
- Self adhesive programming chart detailing how to define which key is simulated by UP, DOWN, LEFT, RIGHT, and FIRE.

- One pack of ten Quick Reference Programming Cards for at-a-glance setting to your games requirements.



- 12 months guarantee and full written instructions.

ONLY 16.95 +£1 P&P

Quickshot II Joystick

NEW IMPROVED GRIP : BUILT IN STABILIZING SUCTION CUPS

TRIGGER FIRE BUTTON : RAPID AUTO FIRE SWITCH : TOP FIRE BUTTON

FROM: MR/MRS/MISS

ADDRESS

SEND C.W.O. (NO STAMP NEEDED) TO: A.G.F. HARDWARE, DEPT.SU.

FREEPOST, BOGNOR REGIS, WEST SUSSEX, PO22 9BR.

QTY	ITEM	ITEM PRICE	TOTAL
	PROGRAMMABLE INTERFACE	27.95	
	JOYSTICK(S)	17.95	
	PACK(S) QUICK REFERENCE CARDS	1.00	
	ZX81 ☐ ZX SPECTRUM ☐ Please tick	FINAL TOTAL	

DEALER ENQUIRIES WELCOME

EXPORT PRICES ON APPLICATION

CENTRONICS INTERFACE

Parallel interface for the Spectrum

Many people want to connect their machines to a better printer than that supplied by Sinclair. Richard Sargent explains the design of a one-chip interface which has extra facilities.

SPECTRUM OWNERS who would like to attach their machines to a printer other than a Sinclair have a choice of commercial interfaces but they are somewhat expensive. If you are intending to make occasional use of a club printer, £40 to £50 is a great deal to tie up in a printer interface. Alternatively, if you are about to buy a printer, £50 is a nasty hidden extra, somewhat worse than VAT. The answer, of course, is to do it yourself, at a cost of less than £10.

The easiest printer interface to make is one which sends eight bits at a time, in parallel, direct to the printer. There is a standard set for it called the Centronics standard, where voltages, signals and pin numbers are all specified. Figure three shows the connections on the Amphenol-type plug which usually is used to plug into the printer. Figure two shows the minimum number of signals which must be exchanged between printer and computer.

The chip chosen to communicate with the printer is the Z-80 parallel input/output port chip, known as the Z80APIO. You will need to obtain the version with the "A", since that is the high-speed version. There are many clever things the chip can do and the technical manual published by Zilog is well worth having if you think you are likely to experiment using the chip.

For our purposes, however, it is its ability to connect to the Spectrum bus with no decoding chips and its ability — on the A port lines — to have some bits configured as inputs and other bits configured as outputs, which makes it such an attractive chip. Figure three shows the connections, with the dashed section being optional.

The prototype was squeezed on to a piece of Veroboard 10cm. by 3.5cm. which did not leave much room for the wiring. The board should not be bigger than 10cm. by 6.5cm., or it will not fit into its neat cassette tape box. The cassette box not only looks neat but protects the circuitry and provides a convenient place for the operating instructions — on the inside of the clear plastic.

Figure four shows the general method of construction. The edge connector should stick out as much as possible, as that permits air to continue to flow from the back of the Spectrum, helping to cool it. A wiring diagram is not relevant — you should list all the connections you need to make and cross each from the list when you have soldered it, double-checking as you go that you have wired to the correct pin.

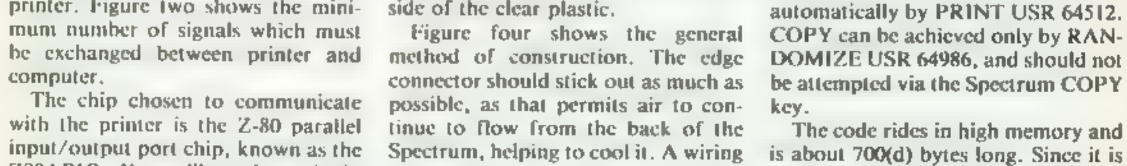
For the cable to the printer I used 12-core flexible cable; that is easy to anchor inside the cassette box but ribbon cable would be equally suitable.

When Sinclair Basic does LPRINT and LLIST it first checks a location in RAM to find the address of a routine in ROM which it will use. That address is 15 bytes after a starting address given by the contents of CHANS. It will move around depending on how many Microdrives are fitted but CHANS is always at 23631. Normally the two bytes of RAM contain 09F4H but they must be changed to FE80H for the Centronics LPRINT and LLIST to work.

NEW and USR 0 both set them back to 09F4H. That setting-up procedure can be done from Basic, or automatically by PRINT USR 64512. COPY can be achieved only by RANDOMIZE USR 64986, and should not be attempted via the Spectrum COPY key.

The code rides in high memory and is about 700(d) bytes long. Since it is assembled for a 48K Spectrum, 16K owners will have to subtract 8000H from all the absolute jumps and calls. So CALL PRINTER CDA3FD be-

Figure 2: One-chip Centronics interface. Computer-printer exchange of signals



CENTRONICS INTERFACE

comes CDA37D. Five other adjustments from FD to 7D are also required , at FD8C, FD36, FD3A, FD60, FD67. Those bytes reference tables.

If space is at a premium only the last 305(d) bytes of code need be entered. That will give the COPY, LPRINT and LLIST but all Spectrum graphics will be printed as spaces and the width of print will be the maximum the printer allows — usually 80 characters. If you can enter only this minimum system, which starts at FDA3H, you must change all four CALL OUTBYTE(CDF3FC) to CALL PRINTIT (CDCCFD). All the bytes down memory of FDA3 give the printer routine a few more facilities.

Many dot matrix printers will output block graphics, though they probably will not have the same codes as the Spectrum graphics. If you patch-in the routine CHUNKY, the Spectrum graphics codes 128-143 will be changed into the codes required to output the small shape of a graphic block on an Epson 80 printer. The routine merely looks up the new code in the table NW so if you have a printer with different codes for the various shapes of graphics, all you need is your own table of codes at NW.

The difficulty with having user-defined graphics and block graphics printed as spaces is that you ought to draw them by hand later. Patch-in the routine NUMBER and they will be printed as their decimal code equivalent, bracketed neatly.

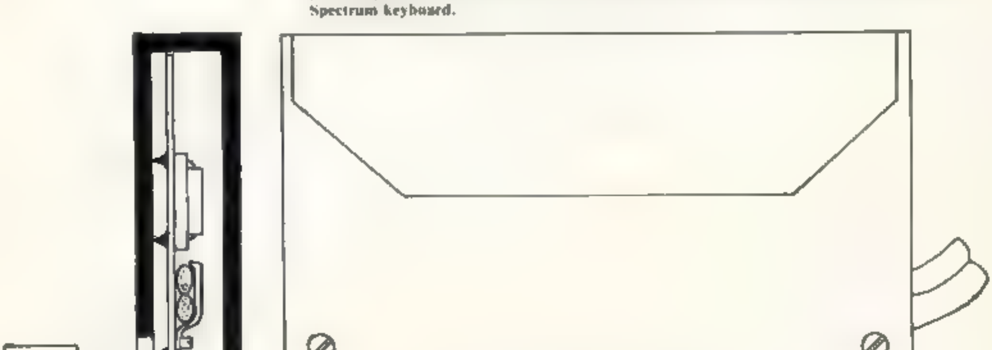
If you wish the print width to be less than the maximum specified by the printer in use, the byte 64753 must be POKED to non-zero and the width, in characters, POKED into 64754.

The hexadecimal dump is not a patch but an extra routine — two routines, in fact. Any area of memory will be sent to the printer from a specified starting address — POKE 64664/5 with the start — to a specified end address — POKE 64666/7 with the finish.

With 64754 set to width 48, RANDOMIZE USR 64668 prints a hexadecimal dump. RANDOMIZE USR 64672 prints the bytes as ASCII but no check is made for control characters and it is recommended that you

Figure 4: One-chip Centronics interface (Spectrum). Mechanical construction.

The Veroboard is held in place by two screws, one of which anchors the cable. The edge-connector is offset to the right, so that the unit, once fitted, does not interfere with the power supply socket of the Spectrum. Note that the copper side of the Veroboard faces the Spectrum keyboard.



CENTRONICS INTERFACE

do not operate this printout unless you are sure your are printing a valid ASCII file.

The copy routine at FDDAH works by printing each pixel dot of the Spectrum screen as a single dot on the printer paper. A good deal of bit-manipulation is necessary to change the format of bits on the screen into the form required by the printer. The maximum size of printout obtainable from the Epson MX70/80/100 series printers is 11cm. wide, 6cm. deep. The width, but not the depth, can be halved by changing the byte at FE50H from 4B to 4C. Doubling the 5.5 × 6cm. image by a software routine would then give 11cm. by 12cm. — interesting but distorted.

The new Epson RX80 printer has a graphics printout, CRT1, which prints normally as 7.5cm, wide by 6cm. deep and is capable of being doubled by software to 16 × 12cm. To do that the routine DOUBLE at FC25H is patched into the main copy routine at location FE2EH. Further bit manipuation is carried-out to print four dots on the paper for every single pixel dot on the screen.

The code may be entered using a hex-loader, and working through the code shown in figure one. If you have an assembler, you should enter the source code and customise the program to suit your requirements.

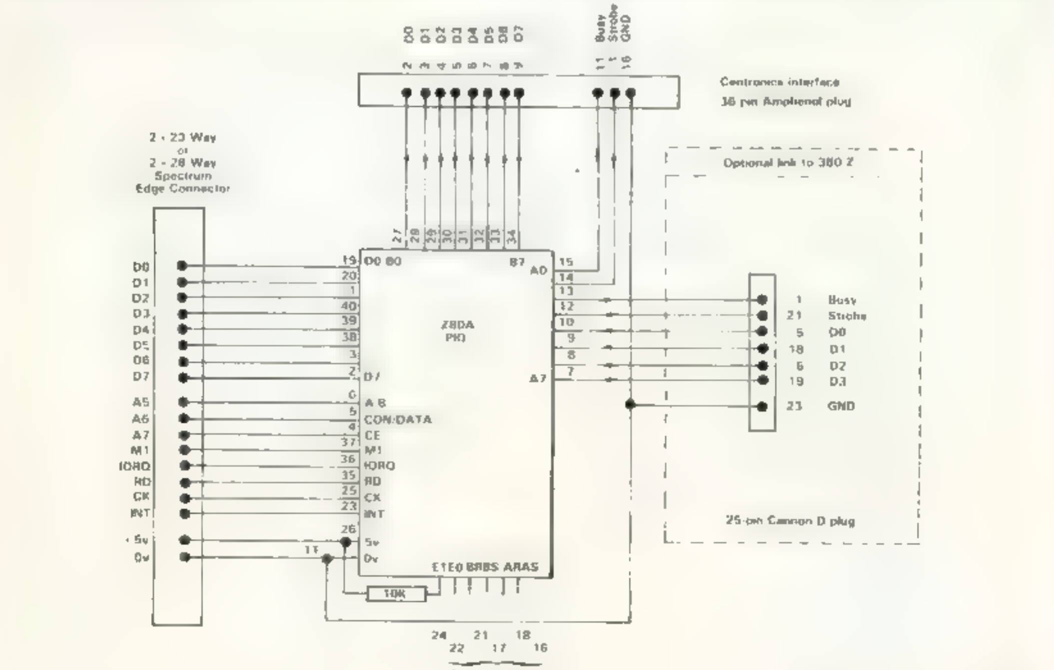
Finally there is the matter of the six spare ports on the PIO chip. They exist, so why not use them? Five have been configured as inputs, so they could be North, South, East, West and Fire signals from a switch-type joystick. They would not, however, be compatible with any commercial software.

There is another rather more intriguing possibility — to use them to communicate with another computer. In many schools the Research Machines 380-Z operates in close proximity to a Spectrum. Often the 380-Z can send only seven bits of parallel data to its printer and therefore cannot print-out graphics. Linking a 380-Z to a Spectrum could not only let the 380-Z print out its graphics but could, with a small amount of software, allow the Spectrum to act as a printer buffer, printing 32K of 380-Z data while the 380-Z was engaged in more productive tasks.

The intended link is one-way only, 380-Z to Spectrum. The PIO chip is configured ready for the link and the hardware requirement is merely the wiring of a Cannon D-type plug, as shown in the wiring diagram.

Two short machine code routines are needed, one for each machine, and the source listings are given. Time has not allowed me to test the code for the 380-Z part of the link but I have a similar link running successfully between a Nascom 2 and a Spectrum. It may be an expensive way to link a Spectrum but for the cost of a piece of cable and a D plug, it is certainly an interesting experiment.

Figure 3: One-chip Centronics interface (Spectrum) circuit diagram with pin nos. The layout of the Spectrum edge connector is given on page 160 of the Spectrum manual.



SAMPLE PRINTOUTS

A COPY of a screen listing.

```
 100 >REM 12345678901234567890123
45678901234567890123456789012345
678901234567890123 45--35digits
 200 LPRINT "hello"
 300 LPRINT "▀▀.▛▜ ▙▄▟▛.▞ █ABC
DEFGHIJKLMNOPQRSTU"
 400 LIST 100
 500 RANDOMIZE USR 64986
9999 STOP
```

A hex dump — line width now set to 48d.

```
F3 AF 11 FF FF C3 CB 11 2A 5D 5C 22 5F 5C 18 43
C3 F2 15 FF FF FF FF FF 2A 5D 5C 7E CD 7D 00 D0
CD 74 00 18 F7 FF FF FF C3 5B 33 FF FF FF FF FF
C5 2A 61 5C E5 C3 9E 16 F5 E5 2A 78 5C 23 22 78
```

Graphics printed as their code numbers.

```
[129][130][131][132][133][134][135][128][142][14
1][140][139][138][137][136][143][144][145][146][
147][148][149][150][151][152][153][154][155][156
][157][158][159][160][161][162][163][164]
```

AS COPY of a screen list, in RX80 CRT graphics, doubled.

```
 10>LPRINT "▀▀.▛▜ ▙▄▟▛.▞ █ABC
DEFGHIJKLMNOPQRSTU"
 100 LIST 10
1000 RANDOMIZE USR 64986
```

```
0131
0131            ;FULL FUNCTION PARALLEL PRINTER INTERFACE
0131            ;R SARGENT July 1983
0131
FC00                        ORG 64512
FC00                        LOAD 64512
FC00
5C4F            CHANS       EQU 23631
0096            TAB1        EQU 150
5B00            BUFFER      EQU 23296
5B00
FC00 2A4F5C     INIT        LD HL,(CHANS)      ;Find vector  FC00H=64512
FC03 110F00                 LD DE,15           ;and load it with
FC06 19                     ADD HL,DE          ;new address
FC07 44                     LD B,H
FC08 4D                     LD C,L
FC09 1180FE                 LD DE,NEWPRINT
FC0C 73                     LD (HL),E
FC0D 23                     INC HL
FC0E 72                     LD (HL),D
FC0F CD88FD                 CALL PIO           ;Initialise PIO chip
FC12 CD6FFE                 CALL CR_LF         ;and clear printer's
FC12                                           ;own buffer
FC15 C9                     RET ;--->To BASIC, vector addr in BC
FC15
FC16 3E1B       E_ROUTINE   LD A,1BH           ;FC16H=64534
FC18 CDCCFD                 CALL PRINTIT
FC1B 3E45                   LD A,"E"
FC1D CDCCFD                 CALL PRINTIT
FC20 3E0D       B_ROUTINE   LD A,0DH           ;FC20H=64544
FC22 C3FFFC                 JP SEND ;and then back into BASIC
FC22
FC22
FC22            ;**************************************
FC22            ;HRG*2 ie: 512 DOTS ACROSS THE PAGE
FC22            ;RX80 CRT GRAPHICS
FC22            ;**************************************
FC22
FC25 C5         DOUBLE      PUSH BC            ;FC25H=64549
FC26 CD3AFC                 CALL PARAMS
FC29 CD7AFC                 CALL T_BUFF
FC2C CD6FFE                 CALL CR_LF
FC2F CD3AFC                 CALL PARAMS
FC32 CD54FC                 CALL B_BUFF
FC35 CD6FFE                 CALL CR_LF
FC38 C1                     POP BC
FC39 [illegible]            RET
FC39
FC3A 3E1B       PARAMS      LD A,1BH
FC3C CDA3FD                 CALL PRINTER
FC3F 3E2A                   LD A,"*"
FC41 CDA3FD                 CALL PRINTER
FC44 3E04                   LD A,4
FC46 CDA3FD                 CALL PRINTER
FC49 3E00                   LD A,00H
FC4B CDA3FD                 CALL PRINTER
FC4E 3E02                   LD A,02H ;0200H bytes sent
FC50 CDA3FD                 CALL PRINTER
FC53 C9                     RET
FC53
FC54 0600       B_BUFF      LD B,0  ;B=256 (Buffer length)
FC56 21005B                 LD HL,BUFFER
FC59 5E         B_LP8       LD E,(HL)     ;Begin to transfer
FC59                                      ;contents of buffer
FC5A CB23                   SLA E
FC5C CB23                   SLA E
FC5E CB23                   SLA E
FC60 CB23                   SLA E
FC62 0E04                   LD C,4        ;Bit manipulation counter
FC64 CB23       B_LP9       SLA E         ;load carry bit from E
FC66 F5                     PUSH AF       ;save carry bit
FC67 CB12                   RL D          ;fill D from carry
FC69 F1                     POP AF        ;repeat
FC6A CB12                   RL D          ;once
FC6C 0D                     DEC C     ;Do 3 more
FC6D 20F5                   JR NZ B_LP9 ;manipulations
FC6F 7A                     LD A,D        ;Ready,
FC70 CDA3FD                 CALL PRINTER  ;print result
FC73 CDA3FD                 CALL PRINTER  ;twice
FC76 23                     INC HL        ;Step along Buffer
FC77 10E0                   DJNZ B_LP8  ;Repeat for whole buffer
FC79 C9                     RET
FC79
FC7A 0600       T_BUFF      LD B,0  ;B=256 (Buffer length)
FC7C 21005B                 LD HL,BUFFER
FC7F 5E         LP8         LD E,(HL)     ;Begin to transfer
FC7F                                      ;contents of buffer
FC80 0E04                   LD C,4        ;Bit manipulation counter
FC82 CB23       LP9         SLA E         ;load carry bit from E
FC84 F5                     PUSH AF       ;save carry bit
FC85 CB12                   RL D          ;fill D from carry
FC87 F1                     POP AF        ;repeat
FC88 CB12                   RL D          ;once
FC8A 0D                     DEC C     ;Do 3 more
```

CENTRONICS INTERFACE

Program continues

```
FC8B 20F5              JR NZ LP9 ;manipulations
FC8D 7A                LD A,D          ;Ready,
FC8E CDA3FD            CALL PRINTER    ;print result
FC91 CDA3FD            CALL PRINTER    ;twice
FC94 23                INC HL     ;Step along Buffer
FC95 10E8              DJNZ LP8   ;Repeat for whole buffer
FC97 C9                RET
FC97
FC97          ;*****************************************
FC97          ;ADDITIONAL ROUTINES - HEX & ASCII DUMP
FC97          ;*****************************************
FC97
FC97          ;(POSIT) MUST BE LOADED WITH START ADDRESS
FC97          ;(FINISH) LIKEWISE WITH END ADDRESS
FC97          ;USE BASIC TO DO THIS
FC97
FC98 0000     START    DW 0        ;Or load from BASIC FC98H=64664
FC9A 4000     FINISH   DW 64       ;Or load from BASIC FC9AH=64666
FC9A
FC9C 0601     HEX      LD B,1              ;FC9CH=64668
FC9E 1802              JR BYTES
FCA0 0600     ASCII    LD B,0              ;FCA0H=64672
FCA2 2A98FC   BYTES    LD HL,(START)
FCA5 222DFD            LD (POSIT),HL
FCA8 AF                XOR A
FCA9 322FFD            LD (PRHEAD),A
FCAC 2A9AFC   LP       LD HL,(FINISH)
FCAF ED5B2DFD          LD DE,(POSIT)
FCB3 B7                OR A
FCB4 ED52              SBC HL,DE
FCB6 2825              JR Z ALLDONE
FCB8 EB                EX DE,HL
FCB9 B0                OR B      ;test HEX or ASCII
FCBA 7E                LD A,(HL)
FCBB 23                INC HL
FCBC 222DFD            LD (POSIT),HL
FCBF 2817              JR Z ASC
FCC1                   ;Process for HEX
FCC1 F5                PUSH AF
FCC2 CB3F              SRL A
FCC4 CB3F              SRL A
FCC6 CB3F              SRL A
FCC8 CB3F              SRL A
FCCA CDE3FC            CALL FORM
FCCD F1                POP AF
FCCE CDE3FC            CALL FORM
FCD1 3E20              LD A,20H
FCD3 CDFFFC            CALL SEND
FCD6 18D4              JR LP
FCD8
FCD8                   ;Process for ASCII
FCD8 CDFFFC   ASC      CALL SEND
FCDB 18CF              JR LP
FCDD
FCDD 3E0D     ALLDONE  LD A,0DH
FCDF CDFFFC            CALL SEND
FCE2 C9                RET ;-->  TO BASIC
FCE2
FCE3 E60F     FORM     AND 0FH
FCE5 FE0A              CP 0AH
FCE7 3802              JR C F2
FCE9 C607              ADD A,7
FCEB C630     F2       ADD A,30H
FCED CDFFFC            CALL SEND
FCF0 C9                RET
FCF0
FCF0          ;*****************************************
FCF0          ;ADDITIONAL CODE -- used by hex/ascii dumps
FCF0          ;may also be used by llist & lprint
FCF0          ;*****************************************
FCF0
FCF1 00       LATCH    DB 0  ;if 0 then width=total width of
FCF1                         ;printer used   FCF1H=64753
FCF2 46       WIDTH    DB 70 ;Eg:70d for A4 paper      FCF2H=64754
FCF2                         ;Eg:48d for a hex dump
FCF3 4F       OUTBYTE  LD C,A              ;FCF3H=64755
FCF4 3AF1FC            LD A,(LATCH)
FCF7 B7                OR A
FCF8 79                LD A,C
FCF9 CACCFD            JP Z PRINTIT;thus CALL OUTBYTE acts
FCF9                               ;as CALL PRINTIT
FCFC C3FFFC            JP SEND     ;thus CALL OUTBYTE acts
FCFC                               ;as CALL SEND
FCFC
FCFF E5       SEND     PUSH HL             ;FCFFH=64767
FD00 6F                LD L,A
FD01 3AF2FC   S1       LD A,(WIDTH)
FD04 3C                INC A
FD05 67                LD H,A
FD06 3A2FFD            LD A,(PRHEAD)
FD09 3C                INC A
FD0A 322FFD            LD (PRHEAD),A
FD0D BC                CP H
FD0E 200E              JR NZ S2
FD10 CD6FFE            CALL CR_LF
FD13 AF                XOR A
FD14 322FFD            LD (PRHEAD),A
FD17 7D                LD A,L
FD18 FE0D              CP 0DH
FD1A 280F              JR Z S4
FD1C 18E3              JR S1
FD1C
FD1E 7D       S2       LD A,L
FD1F FE0D              CP 0DH
FD21 2005              JR NZ S3
FD23 AF                XOR A
FD24 322FFD            LD (PRHEAD),A
FD27 7D                LD A,L
FD28 CDCCFD   S3       CALL PRINTIT
FD2B E1       S4       POP HL
FD2C C9                RET
FD2C
FD2D 0000     POSIT    DW 0 ;For internal use
FD2F 0000     PRHEAD   DW 0 ;For internal use
FD2F
FD2F
FD2F          ;*****************************************
FD2F          ;CODE TO OUTPUT CHARACTER CODE NUMBER
FD2F          ;IN THE FORM [1xx] INSTEAD OF
FD2F          ;A SINGLE SPACE
FD2F          ;*****************************************
FD31
FD31 2600     NUMBER   LD H,0              ;FD31H=64817
FD33 6F                LD L,A
FD34 117BFD            LD DE,RAM
FD37 DD2176FD          LD IX,SUM
FD3B 3E2F     LP2      LD A,2FH
FD3D DD4E00            LD C,(IX+0)
FD40 DD4601            LD B,(IX+1)
FD43 C601     LP3      ADD A,1
FD45 ED42              SBC HL,BC
FD47 F243FD            JP P LP3
FD4A 09                ADD HL,BC
FD4B 12                LD (DE),A
FD4C DD23              INC IX
FD4E DD23              INC IX
FD50 13                INC DE
FD51 0D                DEC C
FD52 20E7              JR NZ LP2
FD53                   ;output code number as [1xx]
FD54 3E5B              LD A,"["
FD56 CDF3FC            CALL OUTBYTE
FD59 3E31              LD A,"1"
FD5B CDF3FC            CALL OUTBYTE
FD5E 3A7DFD            LD A,(RAM2)
FD61 CDF3FC            CALL OUTBYTE
FD64 23                INC HL
FD65 3A7EFD            LD A,(RAM1)
FD68 CDF3FC            CALL OUTBYTE
FD6B C3?7FE            JP E?
FD6E
FD6E 6400?            
FD72 ?A       SUM      DB 100,27H,0E8H,3,64H
FD7? 00 0000?         
FD7? 00                DB 0,0AH,0,1,0
FD7B 000000?  RAM      DB 0,0,0
FD7C 00       RAM2     DB 0
FD7C 00       RAM1     DB 0
FD7C
FD7C          ;*******************************
FD7C          ;PRINT CHUNKY GRAPHICS ROUTINE
FD7C          ;eg:FOR EPSON MX 80
FD7C          ;*******************************
FD7C
FD7C          ;If A not in range 128-143 inc then JP C5
FD7C          ;else load Acc and JP EXIT
FD7C
FD7D FE80     CHUNKY   CP 128              ;FD7DH=64893
FD7F DA8CFE            JP C C5
FD82 FE90              CP 143+1
FD84 D28CFE            JP NC C5
FD84                   ;output block graphics
FD87 1600              LD D,0
FD89 5F                LD E,A
FD8A 2113FD            LD HL,NW-128 ;address of newcode
FD8D B7                OR A
FD8E 19                ADD HL,DE
FD8F 7E                LD A,(HL)
FD90 C3DFFE            JP EXIT
FD90
FD93 A0A2A1A3
FD97 A8AAA9AB NW       DB 0A0H,0A2H,0A1H,0A3H,0A8H,0AAH,0A9H,0ABH
FD9B A4A6A5A7
FD9F ACAEADAF          DB 0A4H,0A6H,0A5H,0A7H,0ACH,0AEH,0ADH,0AFH
FDA3          EXTRAFIN EQU $
01A3          EXTRA    EQU EXTRAFIN-INIT     ;
01A3
01A3          ;*****************************************
01A3          ;This is the start of the code for the
01A3          ;smallest system    [Copy,Llist,Lprint]
01A3          ;
01A3          ;              PRINTER ROUTINE
01A3          ;OUTPUTS :  8 BITS DATA + 1 BIT STROBE
01A3          ;INPUTS  :  1 BIT READY SIGNAL
01A3          ;*****************************************
01A3
005F          P_A_CON  EQU 5FH
007F          P_B_CON  EQU 7FH
001F          P_A_DATA EQU 1FH
003F          P_B_DATA EQU 3FH
003F
FDA3 F5       PRINTER  PUSH AF             ;FDA3H=64931
FDA4 F5                PUSH AF
FDA5 CDB4FD            CALL PIO          ;initialise PIO chip
```

CENTRONICS INTERFACE

```
FDAB DBIN     RDY      IN A,(P A DATA)
FDAA CB47              BIT 0,A
FDAE 20FA              JR NZ RDY
FDAE F1                POP AF
FDAF D3?1              OUT (P B DATA),A
FDB1 3E04              LD A,04H         ;       10
FDB3 D3?1              OUT (P A DATA),A ;strobe sent
FDB5 3E??              LD A,??H         ;       11
FDB7 D3?1              OUT (P A DATA),A ;strobe off
FDB9 F1                POP AF
FDBA C9                RET
FDBB
FDBB 3EFF     PIO      LD A,0FFH        ;A lines as "Control"FDBBH=64955
FDBD D3?3              OUT (P A CONT),A
FDBF 3EF9              LD A,0F9H        ;11111001 Bits 1&2=output
FDC1 D3?3              OUT (P A CONT),A
FDC3 3E0F              LD A,0FH         ;B lines=output
FDC5 D3?3              OUT (P B CONT),A
FDC7 3E06              LD A,06H         ;Strobe off
FDC9 D3?1              OUT (P A DATA),A
FDCB C9                RET
FDCD
FDCC CDA3FD   PRINTLI  CALL PRINTER          ;FDCCH 64972
FDCF FE0D              CP 0DH
FDD1 C0                RET NZ ;    Code 2ol (LNH) if linefeed
FDD1                              ;is Hardwired, else code#FDD1H=64977
FDD1                              ;192 (C0)
FDD2 F5                PUSH AF
FDD3 3E0A              LD A,0AH
FDD5 CDA3FD            CALL PRINTER
FDD8 F1                POP AF
FDD9 C9                RET
FDD9
FDD9                   ;**************************
FDD9                   ;  ROUTINE No 3    COPY
FDD9                   ;**************************
FDD9                   ;RANDOMIZE USR  COPY.
FDD9
FDDA CD75FE   COPY     CALL ESC A           ;FDDAH=64986
FDDD 3E08              LD A,8
FDDF CDA3FD            CALL PRINTER
FDE2 CD6EFE            CALL CR LF
FDE2
FDE5 0600              LD B,0          ;Row 0
FDE7 0E00     COORD1   LD C,0          ;Column 0
FDE9 DD210050          LD IX,BUFFER
FDE9                   ;Address conversion
FDED 79       COORD2   LD A,C          ;Use B & C software counters
FDEE CB3F              SRL A           ;to create a Spectrum
FDF0 6F                LD L,A          ;display file address in
FDF1 78                LD A,B          ;register HL
FDF2 E630              AND 30H         ;00110000
FDF4 0F                RRCA
FDF5 67                LD H,A
FDF6 78                LD A,B
FDF7 E60E              AND 0EH         ;00001110
FDF9 07                RLCA
FDFA 07                RLCA
FDFB 07                RLCA
FDFC 07                RLCA
FDFD B5                OR L
FDFE 6F                LD L,A
FDFF 78                LD A,B
FE00 E601              AND 1           ;00000001
FE02 07                RLCA
FE03 07                RLCA
FE04 B4                OR H
FE05 F640              OR 40H
FE07 67                LD H,A
FE07                   ;HL valid for use in FORM
FE08 C5                PUSH BC
FE09 0608              LD B,8
FE09
FE0B E5       FORM2    PUSH HL
FE0C 1600              LD D,0
FE0F 3E00              LD A,B
FE10 4E       F2B      LD C,(HL)
FE11 58                LD E,B
FE12 CB39     F3B      SRL C
FE14 1D                DEC E
FE15 20FB              JR NZ F3B
FE17 CB12              RL D
FE19 24                INC H
FE1A 3D                DEC A
FE1B 20F3              JR NZ F2B
FE1D 7A                LD A,D
FE1F E1                POP HL
FE1E
FE1F DD7700            LD (IX+0),A
FE22 DD23              INC IX
FE24 10E5              DJNZ FORM2
FE24
FE26 C1                POP BC
FE26
FE26                   ;Do next column
FE26
FE27 0C                INC C
FE28 0C                INC C
FE29 79                LD A,C
FE2A FE40              CP 64
FE2C 38BF              JR C COORD2
FE2C                   ;All columns converted for this Row
FE2C                   ;Send print buffer to printer
FE2E CD44FE            CALL SINGLE (or Call Double    FE2EH=65070
FE2E                   ;Step down a Row
```

```
FE31 04                INC B
FE32 04                INC B
FE33 78                LD A,B
FE34 FE30              CP 48
FE36 38AF              JR C COORD1
FE36
FE36                   ;All done now
FE38 CD75FE            CALL ESC_A
FE3B 3E0C              LD A,12
FE3D CDA3FD            CALL PRINTER
FE40 CD6EFE            CALL CR_LF
FE43 C9                RET; --> To ZX MON
FE43
FE44 C5       SINGLE   PUSH BC              ;FE44H=65092
FE45 00000000
FE49 00                DB 0,0,0,0,0
FE4A 3E1B              LD A,1BH
FE4C CDA3FD            CALL PRINTER
FE4F 3E4B              LD A,"K"      ;Graphic mode K
FE51 CDA3FD            CALL PRINTER
FE54 3E00              LD A,00H      ;0100H bytes being sent
FE56 CDA3FD            CALL PRINTER
FE59 3E01              LD A,01H
FE5B CDA3FD            CALL PRINTER
FE5B
FE5E 210050            LD HL,BUFFER
FE61 0600              LD B,0
FE63 7E       PLP      LD A,(HL)
FE64 CDA3FD            CALL PRINTER
FE67 23                INC HL
FE68 10F9              DJNZ PLP
FE6A CD6EFE            CALL CR LF
FE6D C1                POP BC
FE6E C9                RET
FE6E
FE6F 3E0D     CR LF    LD A,0DH
FE71 CDCCFD            CALL PRINTLI
FE74 C9                RET
FE74
FE75 3E1B     ESC_A    LD A,1BH
FE77 CDA3FD            CALL PRINTER
FE7A 3E41              LD A,"A"
FE7C CDA3FD            CALL PRINTER
FE7F C9                RET
FE7F
FE7F                   ;**********************************
FE7F                   ;  ROUTINES 1 & 2 -- LLIST LPRINT
FE7F                   ;**********************************
FE7F
FE80 F5       NEWPRINT PUSH AF             ;FE80H=65152
FE81 FE7F              CP 127          ;Process copyright code
FE83 2004              JR NZ C4
FE85 3E63              LD A,"c"
FE87 1846              JR EXIT
FE89 000000   C4       DB 0,0,0        ;Process codes 128-143
FE89                                        ;FE89H=65161
FE89                                   ;C4 to hold JP CHUNKY (C3 xx xx)
FE89                                   ;C5 to read CP 144 (90H) if printer
FE89                                   ;is Epson MX80 with block graphics
FE89
FE8C FE90     C5       CP 120          ;Process codes xxx-164
FE8E 3808              JR C C6
FE90 FEA5              CP 164+1
FE92 3007              JR NC C6
FE94 000000            DB 0,0,0        ;or C3 xx xx (JP NUMBER) FE94H=65172
FE97 3E20     C3       LD A,20H        ;code 5D is 1
FE99 1834              JR EXIT
FE99
FE9B FEFF     C6       CP 255          ;Process tokens
FE9D 2808              JR Z RESM
FE9F FEA5              CP 165
FEA1 382E              JR C EXIT
FEA3 FEFF              CP 255
FEA5 302B              JR NC EXIT
FEA7 F5       RESM     PUSH AF
FEA8 3E20              LD A,20H
FEAA CDF3FC            CALL OUTBYTE
FEAD F1                POP AF
FEAE 219600            LD HL,TABL
FEB1 D6A5              SUB 165
FEB3 2808              JR Z R3
FEB5 47                LD B,A
FEB6 CB7E     R2       BIT 7,(HL)
FEB8 23                INC HL
FEB9 28FB              JR Z R2
FEBB 10F9              DJNZ R2
FEBD 7E       R3       LD A,(HL)
FEBE CB7F              BIT 7,A
FEC0 2006              JR NZ R4
FEC2 CDF3FC            CALL OUTBYTE
FEC5 23                INC HL
FEC6 18F5              JR R3
FEC8 CBBF     R4       RES 7,A
FECA CDF3FC            CALL OUTBYTE
FECD 3E20              LD A,20H     ;trailing space
FECD
FECF CDF3FC   EXIT     CALL OUTBYTE
FED2 F1                POP AF
FED3 C9                RET ;--> disappear back into ROM
FED3
FED4          FIN      EQU $
0131          LEN      EQU FIN-PRINTER  ;0131H=00305
0131
0131
```

Figure 1
HEX DUMP

```
FC00 2A 4F 5C 11 0F 00 19 44 4D 11 80 FE 73 23 72 CD
FC10 BB FD CD 6F FE C9 3E 1B CD CC FD 3E 45 CD CC FD
FC20 3E 0D C3 FF FC C5 CD 3A FC CD 7A FC CD 6F FE CD
FC30 3A FC CD 54 FC CD 6F FE C1 C9 3E 1B CD A3 FD 3E
FC40 2A CD A3 FD 3E 04 CD A3 FD 3E 00 CD A3 FD 3E 02
FC50 CD A3 FD C9 06 00 21 00 5B 5E CB 23 CB 23 CB 23
FC60 CB 23 0E 04 CB 23 F5 CB 12 F1 CB 12 0D 20 F5 7A
FC70 CD A3 FD CD A3 FD 23 10 E0 C9 06 00 21 00 5B 5E
FC80 0E 04 CB 23 F5 CB 12 F1 CB 12 0D 20 F5 7A CD A3
FC90 FD CD A3 FD 23 10 EB C9 00 00 40 00 06 01 1B 02
FCA0 06 00 2A 9B FC 22 2D FD AF 32 2F FD 2A 9A FC ED
FCB0 5B 2D FD B7 ED 52 28 25 EB B0 7E 23 22 2D FD 28
FCC0 17 F5 CB 3F CB 3F CB 3F CB 3F CD E3 FC F1 CD E3
FCD0 FC 3E 20 CD FF FC 18 D4 CD FF FC 18 CF 3E 0D CD
FCE0 FF FC C9 E6 0F FE 0A 38 02 C6 07 C6 30 CD FF FC
FCF0 C9 00 46 4F 3A F1 FC B7 79 CA CC FD C3 FF FC E5
FD00 6F 3A F2 FC 3C 67 3A 2F FD 3C 32 2F FD BC 20 0E
FD10 CD 6F FE AF 32 2F FD 7D FE 0D 28 0F 1B E3 7D FE
FD20 0D 20 05 AF 32 2F FD 7D CD CC FD E1 C9 00 00 00
FD30 00 26 00 6F 11 7B FD DD 21 6E FD 3E 2F DD 4E 00
FD40 DD 46 01 C6 01 ED 42 F2 43 FD 09 12 DD 23 DD 23
FD50 13 0D 20 E7 3E 5B CD F3 FC 3E 31 CD F3 FC 3A 7B
FD60 FD CD F3 FC 23 3A 7C FD CD F3 FC C3 97 FE 10 27
FD70 E8 03 64 00 0A 00 01 00 00 00 00 00 00 FE 80 DA
FD80 8C FE FE 90 D2 8C FE 16 00 5F 21 13 FD B7 19 7E
FD90 C3 CF FE A0 A2 A1 A3 A8 AA A9 AB A4 A6 A5 A7 AC
FDA0 AE AD AF F5 F5 CD BB FD DB 1F CB 47 20 FA F1 D3
FDB0 3F 3E 04 D3 1F 3E 06 D3 1F F1 C9 3E CF D3 5F 3E
FDC0 F9 D3 5F 3E 0F D3 7F 3E 06 D3 1F C9 CD A3 FD FE
FDD0 0D C0 F5 3E 0A CD A3 FD F1 C9 CD 75 FE 3E 08 CD
FDE0 A3 FD CD 6F FE 06 00 0E 00 DD 21 00 5B 79 CB 3F
FDF0 6F 7B E6 30 0F 67 7B E6 0E 07 07 07 07 B5 6F 78
FE00 E6 01 07 07 B4 F6 40 67 C5 06 08 E5 16 00 3E 08
FE10 4E 58 CB 39 1D 20 FB CB 12 24 3D 20 F3 7A E1 DD
FE20 77 00 DD 23 10 E5 C1 0C 0C 79 FE 40 38 BF CD 44
FE30 FE 04 04 78 FE 30 38 AF CD 75 FE 3E 0C CD A3 FD
FE40 CD 6F FE C9 C5 00 00 00 00 00 3E 1B CD A3 FD 3E
FE50 48 CD A3 FD 3E 00 CD A3 FD 3E 01 CD A3 FD 21 00
FE60 5B 06 00 7E CD A3 FD 23 10 F9 CD 6F FE C1 C9 3E
FE70 0D CD CC FD C9 3E 1B CD A3 FD 3E 41 CD A3 FD C9
FE80 F5 FE 7F 20 04 3E 63 18 46 00 00 00 FE 80 38 0B
FE90 FE A5 30 07 00 00 00 3E 20 18 34 FE FF 28 08 FE
FEA0 A5 38 2C FE FF 30 28 F5 3E 20 CD F3 FC F1 21 96
FEB0 00 D6 A5 28 08 47 CB 7E 23 28 FB 10 F9 7E CB 7F
FEC0 20 06 CD F3 FC 23 18 F5 CB BF CD F3 FC 3E 20 CD
FED0 F3 FC F1 C9 00 00 00 00 00 00 00 00 00 00 00 00
```

Program 2a

```
                ;380Z TO SPECTRUM ONE-WAY LINK, TO CONNECT
                ;WITH THE PARALLEL PRINTER INTERFACE
                ;R SARGENT July 1983

                P_A_DATA  EQU 1FH

                PRINTIT   EQU 0FD80H ;64896d (48K Spectrum)
                          ;or  7D80H  32128d (16K Spectrum)

                PIO       EQU 0FD6FH ;64879d
                          ;or  7D6FH  32111
                          ;Deduct 32768 to achieve addresses
                          ;for the 16K Spectrum
                          ;Don't forget to adjust the CALLs

F3              L380Z     DI                  ;Interrupt off
CD6FFD                    CALL PIO            ;Set PIO
CD____                    CALL S_N_RDY        ;Send "Not ready"
01FEF7                    LD BC,63486
ED78            WAIT_GO   IN A,(C)            ;Start when
CB47                      BIT 0,A             ;key "1" is
20FA                      JR NZ WAIT_GO       ;pressed
CD____          MLOOP     CALL G_DATA         ;Get 4 bits
CD____                    CALL S_N_RDY        ;Not ready
4F                        LD C,A              ;because
CB39                      SRL C               ;the
CB39                      SRL C               ;bits
CB39                      SRL C               ;are being
CB39                      SRL C               ;shifted
3E80                      LD A,80H
3D              DEL       DEC A               ;Delay
20FD                      JR NZ DEL
CD____                    CALL G_DATA         ;Next 4 bits
E6F0                      AND 0F0H            ;merged with
B1                        OR C                ;first 4
CD____                    CALL S_N_RDY        ;Not ready
CD80FD                    CALL PRINTIT        ;Print 8 bits
18DE                      JR MLOOP            ;go round again


3E02            G_DATA    LD A,2;00000010     ;Send a
D31F                      OUT (P_A_DATA),A    ;"ready" signal
01FEEF                    LD BC,61438
DB1F            STR       IN A,(P_A_DATA)     ;Read 5 bits until

57                        LD D,A
ED78                      IN A,(C)            ;Check if "0"
CB47                      BIT 0,A             ;key pressed
2805                      JR Z ABORT

CB5A                      BIT 3,D             ;strobe received
20F3                      JR NZ STR           ;then exit with
C9                        RET                 ;4 bits valid
```

```
E1          ABORT      POP HL              ;adjust stack
FB                     EI
C9                     RET

F5          S_N_RDY    PUSH AF             ;Send a
3E06                   LD A,6;00000110     ;"not ready"
D31F                   OUT (P_A_DATA),A    ;signal
F1                     POP AF
C9                     RET
```

Program 2b

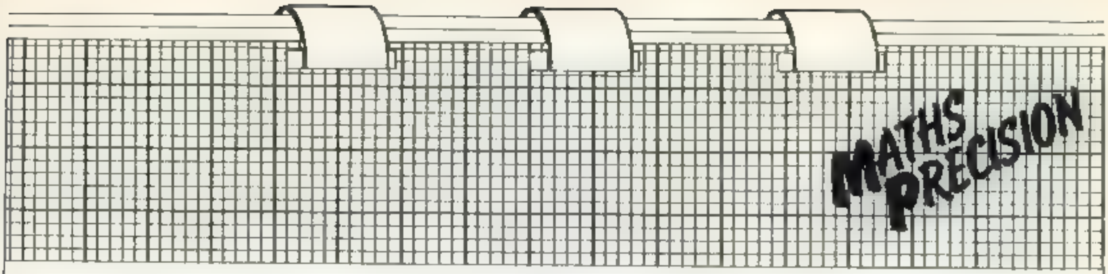
```
;380Z TRANSMIT TO SPECTRUM USING 7 BIT
;I/O PORT  8 BITS OF DATA SENT AS
;TWO SEPARATE NIBBLES  ** PROVISIONAL **

PORT        EQU 0FBFFH
EMT         EQU 0F7H
KBDTC       EQU 1EH
CENTRON     EQU 0000;enter appropriate number
                    ;the number will be found contained
                    ;in locations FF25/FF26H at switch-
                    ;on time

            ORG 1000H
            ;LOAD WHERE-EVER

F5      ENTRY      PUSH AF
F5                 PUSH AF
CD__ __            CALL CENTRON      ;Send low nibble
AF                 XOR A
3D      DY         DEC A             ;Delay
20FD               JR NZ DY
F1                 POP AF
1F                 RRA               ;Shift
1F                 RRA               ;high
1F                 RRA               ;nibble
1F                 RRA               ;and
CD__ __            CALL CENTRON      ;send it
AF                 XOR A
3D      DDY        DEC A             ;Delay
20FD               JR NZ DDY
F1                 POP AF
C9                 RET

AF      RELEASE    XOR A             ;Continue
CD__ __            CALL ENTRY        ;sending
F71E               DB EMT,KBDTC      ;nulls to Spectrum
FE30               CP "0"            ;untill stopped by
20F6               JR NZ RELEASE     ;the "0" key
F700               DB EMT,0          ;Return to Command level
```



Increasing accuracy boosts Spectrum uses

Stephen Rush manipulates strings to perform mathematical functions

ONE OF the problems with the Spectrum, and most other home computers when used for accountancy, astronomy and the like where numbers have to be very precise, is that the Spectrum will display numbers only to eight digits after which either it will ignore the rest of the number — if the number is between 10↑8 and 10↑ 6 — or it follows the number with a letter E and the power of 10.

That unfortunately is often insufficient when performing very complex calculations. Also occasionally, if not very often, numbers bigger than 10↑38 or less than 10↑-38 are required and they not only cannot be held to any real accuracy but also cause the Spectrum to stop with an error report, leaving you to find another way of finishing the problem.

SLICING

The easiest way to circumvent those problems is to use string variables to hold the numbers — they will hold the numbers to any amount of places without the Spectrum trying to change them for you — and manipulate those strings, using the excellent string-slicing capabilities, to perform the four major mathematical operations.

To get down to the programming, please note that the first four subroutines are only to demonstrate the basics and so to make them easier to understand they will work only with positive integers — i.e., no negative numbers and no decimal points — but the final program which contains adapted versions of the programs also contains other routines which allow the user to enter both decimals and negative numbers.

The most obvious and probably easiest program is a simple addition program which will add two strings and leave the answer in a third string. Note that if you enter the multiply program you will need to have this addition routine in the computer as it is an integral part of the multiply procedure.

If you look at figure one you will see that the routine is based on a simple loop which looks at successive elements of each string, adds them, and puts the answer in a third string. The main lines of interest are:

Line 1080, which sets the variable Z to the value of the 'f'th element of the shorter string, after first checking that the string is long enough to allow it.

DIVIDES

Line 1090 adds the 'f'th element of the longer string to Z, adds any remainder from the last run through the loop and then divides the answer by 10.

Figure 1. Add.

```
1000 REM ADD
1010 INPUT "First No.",F$ : INPUT
"Second No.",S$
1020 LET T$ = "" : LET R = 0
1030 LET C$ = "0" + F$ : LET
D$ = "0" + S$
1050 IF LEN S$ > LEN F$ THEN LET
C$ = "0" + S$ : LET D$ = "0" + F$
1060 FOR F = 0 TO LEN C$ - 1
1070 LET Z = 0
1080 IF LEN D$ >= F + 1 THEN LET
Z = VAL D$ (LEN D$ - F)
1090 LET R = (VAL(LEN
C$ - F)) + R + Z)/10
1100 LET T$ = STR$ ((R - INT
R)*10) + T$
1110 LET R = INT R
1120 NEXT F
1140 PRINT T$
```

Line 1100 puts the decimal part — the part after the decimal point — of the number, which corresponds to the units, on to the total (T$) as the next digit of the final answer.

Line 1110 removes the decimal part of the number, the units, thus leaving R as the remainder from the addition.

This routine, although not fast, will add two numbers each 50 digits long in less than 10 seconds, which should not cause too much distress even if you want to add two very large numbers.

The next routine will subtract one string from another, though as we are still dealing only in positive numbers this routine will find the difference between the two numbers. If you want the routine to do a proper subtraction, you should add a line 640 to read:

IF S$>F$ THEN LET T$ = "-" + T$ to the program — figure two.

If you look at the subtraction routine in figure two you will see that, like the addition routine, it is based on a short loop but unfortunately this one is a little more complicated, so I will make its explanation a little more detailed.

COMPARING

Lines 530 and 540 are a simple if rather clumsy way of making both strings the same length by adding "0"s to the beginning of the shorter string.

Lines 560 and 570 set A$ to the larger number and B$ to the smaller number.

Beginners should note that when comparing strings the computer does not compare the values of the strings but instead checks on the codes of the

MATHS PRECISION

first element of each string — or successive elements if the first elements are the same — and the string which begins with the character with the higher code — see appendix A of the Spectrum manual — is the higher string.

Fortunately as the codes of the numbers are stored in numerical order — i.e., the code of "2" is larger than the code of "1" — this has the same effect as comparing the values of the strings. Unfortunately, however, if the strings are of different lengths you may arrive at the incorrect answer.

For example, if the computer compares the two strings "9" and "800", the computer will give the answer that "9" is larger, which is not the correct answer in this instance. To prevent that "0"s have been added to make the strings the same length. So the example changes to "009" and "800" which gives the answer required, i.e., that "800" is larger.

NEXT DIGIT

Line 600 checks if the subtraction is to need a carry from the next digit.

Line 610 does the subtraction — R is the 'carry' if it is needed and R1 is the carry from the last subtraction if there was one — and leaves the result as the next digit of the final answer.

Line 620 then sets R1 to the number which has to be carried forward for the next time round the loop.

The program will find the difference between two numbers each 50 digits long in about 10 seconds.

The next program will multiply two strings by long multiplication. Figure three will show that the program is based on two small loops, one inside the other. The outside loop goes through one of the strings, one element at a time, and the inside loop multiplies that number by each element of the second loop in turn. Note that to use the multiplication routine you must have the addition routine in the computer or the multiplication program will not work.

Line 140 sets the remainder (R) to zero and sets the subtotal (Z$) to the empty string.

Line 160 multiplies the correct elements of the strings and adds any remainder from the last multiplication.

Line 170 puts the units part of the calculation on to Z$ as the next digit of the subtotal.

Line 180 calculates the remainder to be carried forward and puts the remainder in the variable R.

Lines 200 and 210 add "0"s to the subtotal and then send the subtotal and the current total to be added using the addition routine.

PURPOSE

For those who do not see the reason for doing this, here is a small example of a long multiplication:

If you want to multiply 12,654 by 124,	12,654 124*
you first multiply 12,654 by 4 giving	50,616
then you multiply 12,654 by 2 and add a "0"	253,080
then multiply it by 1 and add "00" giving	1,265,400
and then to get the final answer you add these three subtotals giving	1,569,096

That is, in effect, the purpose of lines 200 and 210, as they add "0"s and then send the strings to be added. The only difference is that those lines add the subtotals after each is calculated rather than waiting to the end and then adding them all.

If you wish to multiply two numbers each 50 digits long, unfortunately this routine will take almost 12½ minutes to arrive at the answer because to do the calculation requires:

A loop of 1 to 50 multipyling each digit by one digit from the other string; a loop of up to 1 to 50 adding "0"s to the subtotal; a loop of between 1 to 50 and 1 to 100 to add the total and the subtotal.

To complete the sum it will have to go through the process 50 times, which unfortunately takes a long time to complete. For many applications the answer is well worth the wait for the accuracy alone but you should also note that the biggest number the Spectrum can normally handle without stopping with an error is easily surpassed by multiplying two numbers each only 20 digits long.

DENOMINATOR

The next routine is slightly limited in that the number by which you are dividing —the denominator — may be only eight digits long and unfortunately that applies to the main program as well as the short routine. If anyone knows of a way of circumventing this, I would love to know how.

The number into which you are dividing — the numerator — can be as long as needed and you also have

Figure 2. Subtract.

```
500 REM SUBTRACT
510 INPUT "First No.",F$,"Second
No.",S$
520 LET T$ = ""
530 IF LEN F$>LEN S$ THEN LET
S$ = "0" + S$ : GOTO 530
540 IF LEN S$>LEN F$ THEN LET
F$ = "0" + F$ : GOTO 540
550 LET R1 = 0
560 LET A$ = "0" + S$ : LET
B$ = "0" + F$
570 IF F$>S$ THEN LET A$ = "0" + F$
: LET B$ = "0" + S$
580 FOR F = LEN A$ TO 1 STEP - 1
590 LET R = 0
600 IF (VAL B$(F) + R1)>VAL A$ (F)
THEN LET R = 10
610 LET T$ = STR$ ((VAL
A$(F) + R) - (VAL B$(F) + R1)) + T$
620 LET R1 = INT (R/10)
630 NEXT F
650 PRINT T$
```

Figure 3. Multiply.

```
98 REM To use this program you must
have the addition routine in the computer
100 REM MULTIPLY
110 INPUT "First No.",F$, "Second
No.",S$
120 LET T$ = "" : LET B$ = S$ : LET
A$ = "0" + F$
130 FOR G = LEN B$ TO 1 STEP - 1
140 LET R = 0 : LET Z$ = ""
150 FOR F = LEN A$ TO 1 STEP - 1
160 LET A = (VAL B$(G)*VAL A$
(F)) + R
170 LET Z$ = STR$(((A/10) - (INT(A/
10)))*10) + Z$
180 LET R = INT (A/10)
190 NEXT F
200 FOR H = 1 TO LEN B$ - G : LET
Z$ = Z$ + "0" : NEXT H
210 LET F$ = Z$ : LET S$ = T$ : GOSUB
1020
220 NEXT G
250 PRINT T$
260 STOP
1130 RETURN
```

the option of choosing to how many places you want the answer worked.

What is probably a much bigger blow to people who will not bother to type-in the main program but want to see the division routine in action is that we are still dealing only in integers and consequently the answer will be a string of numbers and you will have to find where to put the decimal point — i.e., "10" divided by "40" will give the answer "0025" instead of "0.25".

If you look at figure four you will see that the routine is very short and the only lines of interest are those between 1565 and 1590.

DECIMAL

The easiest way to describe the tasks of those lines is to show an example of a long division next to a description of how the computer does the calculation. For example, if you wanted to divide 9,714 by 12 to five places, in practice you would PRINT 9714/12 but showing an example to 20 places would take too much room and would also put my one typing finger out of action for the rest of the week. The calculation would be:

Line	Action	The division
		08095
1510	Sets F$ to "9714" and S$ to "12".	12)9714
1565	Sets Z to 9.	9
1580	Divides 9 by 12 and puts the answer (0) on the total.	
1590	Calculates the remainder (9).	
1565	'Brings the 7 down'.	97
1570	Sets R to the number formed (97).	
1580	Divides 97 by 12 and puts the answer (8) on the total.	97 (12*8)
1590	Finds the remainder (1).	1
1565	'Brings the 1 down'.	11
1570	Sets R to the number formed (11).	
1580	Divides 11 by 12 and puts the answer (0) on the total.	11_ (12*0)
1590	Finds the remainder (11).	11
1565	'Brings the 4 down'.	114
1570	Sets R to the number formed (114).	
1580	Divides 114 by 12 and puts the answer (9) on the total.	114_ (12*9)
1590	Finds the remainder (6).	6
1565	Finds that F$ is too short so it 'brings down a 0 down'.	60
1570	Sets R to the number formed (60).	
1580	Divides 60 by 12 and puts the answer (5) on the total.	
1620	Arrives at the final answer	08095

As you can see, the decimal point still needs to be placed and the easiest way of doing it is to PRINT VAL F$/VAL S$, which will give an approximate value which should help you to place the point. This routine will do a division to 1,000 places in about three minutes.

As you have probably already said to yourselves, the routines are almost useless on their own as much of the calculations you will want to do involves negative numbers and/or numbers with decimal points.

To circumvent those problems requires two more routines. The first — lines 9500 to 9550 of the main program, figure five — searches the two strings — F$ and S$ — to find any decimal points. When it finds a decimal point it removes it and remembers from where in the string it came. The routine sets DP1 to the number of digits after the decimal point in F$ and DP2 to the number of digits after the decimal point in S$.

The other routine, from 9650 to line 9600, checks if the first character of F$ or S$ is a negative sign; the variable NEG is set to 0 if both numbers are positive, to 1 if the first is negative and the second positive, to 2 if the first is positive and the second negative, and to 3 if both numbers are negative.

CHECKING

The final routine, though not very important, is very useful in tidying the answers; its job is to remove the trailing and preceding zeros which make the answers look ugly. The important lines of this final routine are:

Line 9930 removes preceding zeros from the variable T$.

Line 9920 returns if the number has been reduced to "0".

Line 9940 is a complicated line which does a fairly simple job; its task is to search the string for a decimal point; if it finds a point it continues with line 9950 but if the string does not contain a decimal point it will return, after first checking whether the string should be negative. This line prevents the rest of the routine removing zeros from a number greater than 1 as they do not need the zeros removing, e.g., you do not want the zeros removed from 9500.

Line 9965 will then replace a zero before the number if it starts with a decimal point — i.e., ".45" will be changed to "0.45". The results from the first two routines are used in the following ways:

For adding, if the first number is positive and the second negative, line 7520 sends the strings to be subtracted. If, however, the first number is negative and the second positive, line 7530 swaps F$ and S$ and also swaps DP1 and DP2 before sending them to be subtracted.

Lines 7560 and 7565 ensure that both numbers have the same number of decimal places by adding "0"s to the string with fewer places after the point.

Line 7670 finds the position of the point in the final answer. When you add two numbers each with x digits after the point the answer will also have x digits after the point and will also make the answer negative if you were adding two negative numbers — i.e., if NEG 3.

To subtract, if the first number is positive and the second negative you are effectively adding two numbers, so line 8040 sends the strings to be added; similarly if the first number is negative and the second positive you are adding the two negative numbers, so line 8030 sends the strings to be added.

Lines 8050 and 8060 ensure that

Figure 4. Divide.

```
1500 REM DIVIDE
1510 INPUT "First No.",F$,"Second
No.",S$
1520 INPUT "No. of places",P
1530 LET T$ = ""
1550 LET R = 0
1560 FOR F = 1 TO P
1565 LET Z = 0 : IF LEN F$>F THEN
LET Z = VAL F$ (F)
1570 LET R = R*10+Z
1580 LET T$ = T$ + STR$ (INT(R/VAL
S$))
1590 LET R = R - (INT(VAL T$(LEN
T$))*VAL S$)
1600 NEXT F
1620 PRINT T$
```

MATHS PRECISION

both numbers have the same number of decimal places.

Line 8190 then finds the position of the decimal point, using the same reasoning that let line 7670 position the point for the add routine.

To multiply, line 8640 makes the final answer negative if one of the numbers was positive and the other negative. This line also places the decimal point for you — when you multiply a number with x digits after the point by a number with y digits after the point the answer will have x + y digits after the point.

To divide, line 9090 finds approximately where the decimal point will be in the final answer; the number of places before the point is roughly the difference in the numbers of places before the points in the original numbers.

ANSWER

Line 9100 allows for people dividing by numbers less than 1.

Line 9120 checks if the second number is larger than the first, in which case the answer will have one fewer digit before the point.

Finally line 9130 works out the final answer miraculously.

When you have typed-in the program you will see that the lines from 100 to 7000 have been left blank and, as you may have guessed, that is where you put your problems. They should take the form LET F$ "First No." : LET S$ - "Second No." followed by one of the following:

```
GOSUB ADD
GOSUB SUBTRACT
GOSUB MULTIPLY
GOSUB DIVIDE
```

depending on how you wish the strings to be manipulated.

Also, if you require them, you have 17 memories at your disposal — all of the string variables except those used in the program. Do not use A$, B$, C$, D$, F$, R$, S$, T$, or Z$. For example, LET M$ = T$ will put the result of the last calculation in memory M$. Before using those memories, however, you should check that you need to store the number, because remembering very large numbers will use up a great deal of your precious RAM — the program takes about 3.1K and another 300 bytes are used by the minimum variables necessary for the program to run. That, of course, will increase considerably when you have to add your own lines.

DO NOT RUN

If, however, you are unfortunate enough to get error 4 OUT OF MEMORY then do not use RUN or CLEAR as they will erase the program variables. Instead you should do one of the following:

Look through the routine you are using and see which string variables are not used and set them to the empty string — e.g., Z$ is not used in the ADD routine, so if you run out of memory while using the routine LET Z$ = "" followed by CONTINUE may help.

The following lines may be removed as they are not important to the running of the program — 1, 7500, 8000, 8500, 9000, 9500, 9650, 9900, and also line 2 if you do not mind sacrificing my vanity.

If you are still short of memory the only solution is to start cannibalising the routines you are not using and then reLOAD the program before using it again.

The main use of the program will be for normal +, −, *, /, and here is an example which will divide 97402.000055 by 98.64 and will then add 65 — to do this sum you should add the following lines to the main problem:

```
100 LET F$ = "97402.000055"
110 LET S$ - "98.64"
120 GOSUB DIVIDE
130 LET F$ = T$
140 LET S$ "  65"
150 GOSUB ADD
```

Another use of the program is to find high powers of numbers by multiplying repeatedly by the same number. For example, to find the exact value of 17.64 to the power of 35, lines 100 to 150 from above should be removed and the following lines should be added:

```
100 LET F$ = "17.64"
110 FOR Y = 2 TO 35
120 LET S$ = "17.64"
130 GOSUB MULTIPLY
140 LET F$ = T$
150 NEXT Y
```

Once this program is running satisfactorily I suggest you go for a stroll to the local shops and buy a paper and a box of tea bags as the program is very slow to reach the exact answer.

Remember when using FOR NEXT loops that you must not use the variables used in the main program as the loop counter — i.e., do not use P, F, Z, R, G, A, or H as the loop counter.

Another function which can be performed easily using this program, but is not practical without it, is a factorial routine. The factorial of a number can be found by multiplying all the numbers between 1 and the number together, i.e., 6 factorial (6!) 6*5*4*1*3*2*1)

SMALL ROUTINE

To find the factorials of 100, 69, and 30 the following lines should be added to the main program:

```
100 LET F$  "2"
110 FOR Y = 3 to 100
120 LET S$ - STR$ Y
130 GOSUB MULTIPLY
140 LET F$ = T$
150 IF Y - 30 THEN PRINT "30
Factorial is "; T$
160 IF Y - 69 THEN PRINT "69
Factorial is "; T$
170 NEXT Y
180 PRINT "100 Factorial is ";
```

Before finally leaving Spectrum owners I would like you all to try to write a small routine — the ones you write are always the best — which will round the last digit of the total — i.e., it will round 0.6666 up to 0.667 and will round 8.3422 down to 8.342 — which may make the final answer more meaningful. Without the routine 10/3*3 will give the answer 9.999 but with the routine the 9.999 will be rounded-up to the proper 10.

If you have not already done so, try to write this routine on a piece of paper — or on the computer if you find it easier to concentrate when sitting in front of the keyboard. If your routine works it will show that you understand at least the basics of the program and should be able to adapt the routine if your particular problem needs the program altering

MATHS PRECISION

in some way. I suppose I had better show one routine which will do the job for those who could not bother and for those who could not get their routines to work properly:

11 LET ROUND = 7450 : LET ROU = 0
7450 REM ROUND
7460 LET F$ = T$: LET S$ = "5" : GOSUB DP
7465 LET DP2 = DP1 : LET NEG = NEG*3 : LET ROU = 1 : GOSUB 7550
7470 IF T$ (LEN T$) = "." THEN LET T$ = T$ (TO LEN T$ − 1)
7475 LET T$ (LEN T$) = "0"
7480 GOSUB ZR
7490 PRINT "The answer after rounding is", T$
7675 IF ROU = 1 THEN LET ROU = 0 : RETURN

And using GOTO ROUND will round the number in T$.

The program will go almost straight on to a ZX-81 with only a few minor changes, due to the multi-statement lines on the Spectrum. Many of the lines can just be split into many separate lines, one after another. The only lines which will require any thought are those which contain IF THEN statements.

When translating them you must remember that if the statement is false — e.g., if NEG < >2 for line 7520 — then the rest of the line is ignored. Most of the lines with an IF THEN statement, and also more than one statement, will therefore require a GOSUB and RETURN or a GOTO. An example is line 7560 which could be replaced by:
7560 IF DP1 > DP2 THEN GOTO 9700
9700 LET S$ = S$ + "0"
9701 LET DP2 = DP2 + 1
9702 GOTO 7650

This approach should help with all of the problem lines except line 9940, which must be replaced by:
9940 FOR F = 1 TO LEN T$
9941 IF T$ (F) = "." THEN GOTO 9950
9942 NEXT F
9943 LET T$ = ("-" and NEG = 5) + T$
9944 RETURN

You should be able to convert the rest of the program using those hints.

If you are still a ZX-80 owner, you will probably have to try to think how the routines work and try to adapt them, remembering that the ZX-80 does numbers only as integers — the strings will still hold decimal numbers but the problem is getting the correct numbers into the strings in the proper order. If anyone wishes to try to adapt the program I wish them the best of luck but from what I remember of the ZX-80 it is probably easier for you to write your own program, using any ideas from my program which help you with the task.

```
   1 REM  PRECISION
   2 REM S. RUSH 1983
   9 INPUT "NO. OF PLACES (FOR D
IVISION)",P
  10 LET T$="": LET ADD=7500: LE
T SUBTRACT=8000: LET MULTIPLY=85
00: LET DIVIDE=9000: LET DP=9500
: LET ZR=9900
  15 REM the problems should go
between lines 20 and 7000
  20 LET f$="2675": LET s$="12":
 GO SUB divide
  25 PRINT t$
  30 LET f$=t$: LET s$="-98": GO
 SUB add: PRINT t$: LET f$=t$: L
ET s$="-16": GO SUB subtract: PR
INT t$: LET f$=t$: LET s$="72":
GO SUB multiply: PRINT t$: LET f
$=t$: LET s$="4": GO SUB divide
7400 PRINT "The answer is";T$
7499 STOP
7500 REM  ADD
7510 GO SUB DP
7520 IF NEG=2 THEN  LET NEG=0: G
O TO 8020
7530 IF NEG=1 THEN  LET R$=F$: L
ET F$=S$: LET S$=R$: LET DP3=DP1
: LET DP1=DP2: LET DP2=DP3: LET
NEG=0: GO TO 8020
7540 LET MULT=0
7550 LET T$="": LET R=0
7560 IF DP1>DP2 THEN  LET S$=S$+
"0": LET DP2=DP2+1: GO TO 7560
7565 IF DP2>DP1 THEN  LET F$=F$+
"0": LET DP1=DP1+1: GO TO 7565
7570 LET C$="0"+S$: LET D$="0"+F
$
7575 IF LEN F$>LEN S$ THEN  LET
C$="0"+F$: LET D$="0"+S$
7580 FOR F=0 TO LEN C$-1
7590 LET Z=0
7600 IF LEN D$>F+1 THEN  LET Z=V
AL D$(LEN D$-F)
7610 LET R=(VAL (C$(LEN C$-F))+R
+Z)/10
7620 LET T$=STR$ ((R-INT R)*10)+
T$
7630 LET R=INT R
7640 NEXT F
7650 IF MULT=1 THEN  LET MULT=0:
 RETURN
7670 LET T$=("-" AND NEG=3)+T$(
TO LEN T$-DP1)+"."+T$(LEN T$-DP1
+1 TO )
7680 GO TO ZR
8000 REM  SUBTRACT
```

MATHS PRECISION

```
8010 GO SUB DP
8020 LET T$=""
8030 IF NEG=1 THEN  LET NEG=3: GO TO 7540
8040 IF NEG=2 THEN  LET NEG=0: GO TO 7540
8050 IF DP1<DP2 THEN  LET F$=F$+"0": LET DP1=DP1+1: GO TO 8050
8060 IF DP2<DP1 THEN  LET S$=S$+"0": LET DP2=DP2+1: GO TO 8060
8070 IF LEN F$>LEN S$ THEN  LET S$="0"+S$: GO TO 8070
8080 IF LEN S$>LEN F$ THEN  LET F$="0"+F$: GO TO 8080
8090 LET R1=0
8110 IF F$>S$ THEN  LET A$="0"+F$: LET B$="0"+S$: GO TO 8130
8120 LET A$="0"+S$: LET B$="0"+F$: LET NEG=3-NEG
8130 FOR F=LEN A$ TO 1 STEP -1
8140 LET R=0
8150 IF (VAL B$(F)+R1)>VAL A$(F) THEN  LET R=10
8160 LET T$=STR$ ((VAL A$(F)+R)-(VAL B$(F)+R1))+T$
8170 LET R1=INT (R/10)
8180 NEXT F
8190 LET T$=("-" AND NEG=3)+T$( TO LEN T$-DP1)+"."+T$(LEN T$-DP1+1 TO )
8200 GO TO ZR
8500 REM  MULTIPLY
8510 GO SUB DP: LET T$="": LET B$="00"+S$: LET A$="00"+F$
8530 FOR G=LEN B$ TO 1 STEP -1
8535 LET R=0: LET Z$=""
8540 FOR F=LEN A$ TO 1 STEP -1
8550 LET A=(VAL B$(G)*VAL A$(F))+R
8570 LET Z$=STR$ (((A/10)-(INT (A/10)))*10)+Z$
8580 LET R=INT (A/10)
8590 NEXT F
8600 FOR H=1 TO LEN B$-G: LET Z$=Z$+"0": NEXT H
8610 LET F$=Z$: LET S$=T$: LET T$="": LET R=0
8620 LET MULT=1: GO SUB 7570
8630 NEXT G
8640 LET T$=("-" AND (NEG=1 OR NEG=2))+T$( TO LEN T$-DP1-DP2)+"."+T$(LEN T$-DP1-DP2+1 TO )
8650 GO TO ZR
9000 REM  DIVIDE
9010 GO SUB DP: LET Z$=""
9030 LET R=0
9040 FOR F=1 TO P
9045 LET Z=0: IF LEN F$>=F THEN  LET Z=VAL F$(F)
9050 LET R=R*10+Z
9060 LET Z$=Z$+STR$ (INT (R/VAL S$))
9070 LET R=R-(INT (VAL Z$(LEN Z$))*VAL S$)
9080 NEXT F
9085 PRINT Z$
9090 LET DP3=(LEN F$-DP1)-(LEN S$-DP2)+1
9100 FOR F=1 TO LEN S$: IF CODE S$=CODE "0" THEN  LET DP3=DP3+1: LET S$=S$(2 TO ): LET Z$="0"+Z$: NEXT F
9110 FOR F=1 TO LEN Z$: IF CODE Z$=CODE "0" THEN  LET Z$=Z$(2 TO ): NEXT F
9120 IF S$>F$ THEN  LET DP3=DP3-1
9125 IF DP3<1 THEN  LET Z$="0"+Z$: LET DP3=DP3+1: GO TO 9125
9130 LET T$=("-" AND (NEG=1 OR NEG=2))+Z$( TO DP3)+"."+Z$(DP3+1 TO )
9140 GO TO ZR
9500 REM  DP
9510 LET DP1=0: LET DP2=0
9520 FOR F=LEN F$ TO 1 STEP -1: IF F$(F)="." THEN  LET DP1=(LEN F$-F): LET F$=F$( TO F-1)+F$(F+1 TO ):
9530 NEXT F
9540 FOR F=LEN S$ TO 1 STEP -1: IF S$(F)="." THEN  LET DP2=(LEN S$-F): LET S$=S$( TO F-1)+S$(F+1 TO )
9550 NEXT F
9560 REM  NEGATIVE
9570 LET NEG=0
9580 IF CODE F$=CODE "-" THEN  LET NEG=1: LET F$=F$(2 TO )
9590 IF CODE S$=CODE "-" THEN  LET NEG=NEG+2: LET S$=S$(2 TO )
9600 RETURN
9900 REM  ZR
9910 IF CODE T$=CODE "-" THEN  LET NEG=5: LET T$=T$(2 TO )
9920 IF T$="0" OR T$="0." THEN RETURN
9930 IF CODE T$=CODE "0" THEN  LET T$=T$(2 TO ): GO TO 9920
9940 FOR F=1 TO LEN T$: IF T$(F)<>"." THEN  NEXT F: LET T$=("-" AND NEG=5)+T$: RETURN
9950 IF T$=".0" THEN  LET T$="0": RETURN
9960 IF T$(LEN T$)="0" THEN  LET T$=T$( TO LEN T$-1): GO TO 9950
9965 IF CODE T$=CODE "." THEN  LET T$="0"+T$
9970 IF NEG=5 THEN  LET T$="-"+T$
9980 RETURN
```

INPUT/OUTPUT PORT FOR ZX SPECTRUM

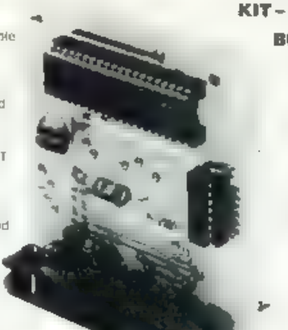
- 3 Channels (24 lines)
- Programmable mix of input and output
- May be used with printer
- Uses IN/OUT instructions
- Up to 8 may be connected

KIT – £16.95

BUILT – £18.95

(Output connector £3.25 if required)

Includes VAT and Inland Postage



Ideal for controlling models, motors, robots, automatic equipment, security systems etc. practical applications circuits included.

Export postage (surface) £2.50. Send SAE for full catalogue of Spectrum and ZX81 accessories. Cash with order or ACCESS

REDDITCH ELECTRONICS

21 FERNEY HILL AVENUE,
REDDITCH, WORCS B97 4RU

hobbyboard

The Complete Printed Circuit Workshop

Computer Cables & Connectors

We now offer an extensive range of computer cables & connectors including a spectrum user port extender cable. Send for complete cable & connector price list.

SPECTRUM EXTENDER

HB/2069 £8.85*

Overcomes those interconnection problems! Six inch double ended extender & male PCB plug. Also available as single ended with 9" of ribbon cable **HB/2068 £4.76***

VIC/CBM 64 cassette interface

Operates with most data recorders HB/3000 £12.50*

Full Hobbyboard Catalogue £1.50* (refundable with order over £10)
*Prices inclusive of VAT, carriage 60p in U.K. Overseas orders please add extra carriage

hobbyboard
complete P.C.B. workshop

a division of
KELAN ENGINEERING Ltd
Hookstone Park
Harrogate, N. Yorks

ZX PROGRAMMERS. . .™
look no further than OCP

Whether you write MACHINE CODE or BASIC we have the very latest "state of the art" programming tools for you. Try them and see why our product is widely regarded by professionals as the very best available.

FULL SCREEN EDITOR/ASSEMBLER (16/48K) voted THE MOST POWERFUL MACHINE CODE PROGRAMMING TOOL YET SEEN by HOME COMPUTER WEEKLY

- Editing facilities comparable to the most sophisticated word processor with MOVE, COPY and/or DELETE lines or blocks of code.
- LOCATE, CHANGE or DELETE specified strings or characters, full Z80 instruction set supported, comprehensive syntax check, powerful expression evaluator, derivatives including stand alone and quash, symbol table display, assembly to screen, printer or save to tape and **"SNAKE"**, a fully notated source code demonstration program
  (PLUS 80 version for the KEMPSTON CENTRONICS 80 COLUMN PRINTER INTERFACE now available).

MACHINE CODE TEST TOOL (16/48K) tutor and de-bug program, co-resides with the FULL SCREEN EDITOR/ASSEMBLER in 48K to give a **COMPLETE MACHINE CODE DEVELOPMENT ENVIRONMENT that is second to none.** The programmer can switch between programs within moments.

- Allows ease entry and testing of machine coded instructions.
- Pages and displays memory registers so you actually see what's happening, displays Main and Alternate register sets. Breakpoints can be Set, Viewed and Nullified. HEX/DECIMAL conversion. STOP and return to BASIC. MOVE a memory block. GOTO address. Character Generator and full supporting documentation and tutorial.

MASTER TOOLKIT (16/48K). YOUR BASIC WILL NEVER BE THE SAME AGAIN!
This program adds a whole range of really powerful commands and facilities for your Spectrum.

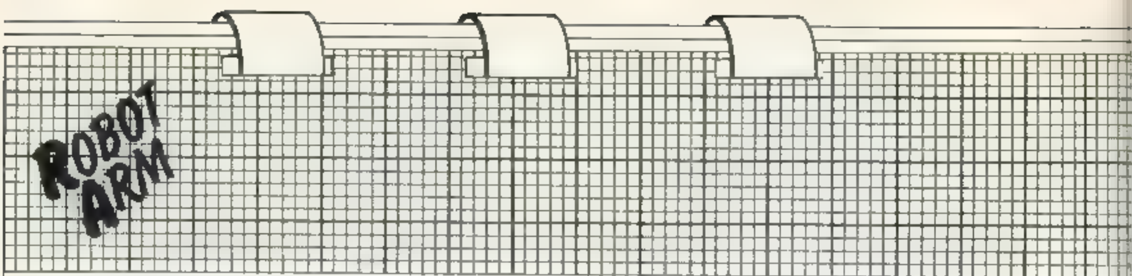
- Real time clock and alarm with off/on/set and print commands.
- BLOCK MOVE/COPY/DELETE and MERGE two lines. FIND and CHANGE character string. RENUMBER. 10 programmable keys (up to 255 chars each). TRACE with continuous execution display. VARIABLE display and dump. COMPRESS, REMKILL and PACK to minimise program bytes. CHANGE CASE upper to lower and vice versa. RAMTOP ADDRESS and PRINTER output for vectors. Comprehensive manual and instructions supplied.

Available from selected branches of W H SMITH, BOOTS and MENZIES and other good software stockists

OCP™ SOFTWARE . . . SIMPLY THE BEST

If you experience any difficulty obtaining your copy of these programs send a cheque or postal order for £9.95 per program (£19.95 for the PLUS 80 version of FULL SCREEN EDITOR/ASSEMBLER)

Oxford Computer Publishing Ltd,
4 High Street, Chalfont St. Peter, Bucks. SL9 9QB



Cheap arm demonstrates robot capabilities

Stephen Crawford shows how by using existing materials and a ZX-81 you can obtain an idea of how large industrial concerns could develop in the years to come

ROBOTS are great fun but they are expensive. Those used by British Leyland cost tens of thousands of pounds while many cheaper designs aimed at schools, colleges and the enthusiast cost anything between £700 and £1,000. Those prices are clearly outside the range of the average pocket. The robot described here aims to rectify that situation for £25 to £30; excluding the ZX-81 and an I/O board, a small and simple robotic arm can be constructed.

Although this robot has no practical use, it serves as an excellent demonstration tool or it can be built for pleasure. The materials required, apart from a ZX-81 and an I/O board, are two Acom radio control servos, some scraps of thin plywood or card, a suitable box and some old Meccano, or similar, rods and wheels.

The construction method need not be adhered to and the robot should be adapted to suit the individual and his materials. First the arm is constructed. The pattern of the arm — figure one — is copied on to 1/32in. plywood or thick card. The arm is cut out and folded along the dashed lines into a box shape. That is then taped or glued. Once the arm is complete, two holes are made as indicated in figure one. The holes are made using either a knitting needle or a special punching tool.

The next step involves making a platform on to which the arm and servo motors are attached. The platform — figure two — is cut from a piece of ⅛in. plywood. Two Meccano brackets — figure three — are then bolted on to the platform as indicated in figure two. If Meccano is unavailable, similar items can be made from thin aluminium.

A 1in. Meccano bush wheel is placed inside the arm so that the centre of the wheel is aligned with the two holes in the arm. The whole arm assembly is placed between the brackets and a 1½in. Meccano rod inserted through the holes. The bush wheel is then glued to the arm and the grub-screw in the bush is tightened, fixing the arm and rod together solidly — figure four.

The next stage involves the mounting of the servo. A gearwheel is fixed on to the right end of the rod, looking from behind. The purpose of the wheel is to provide a large surface area on to which a servo head may be attached. The servo is then mounted on to the platform. The centre of the servo head must be aligned accurately with the centre of the rod. To do that the servo must be raised to the correct height.

Small blocks of wood glued to the platform act as supports. The servo is attached, both to the gearwheel and the platform, by double-sided tape. It should be noted that the bolt securing the servo head to the servo should be removed, as it causes a bump which weakens the joint — figure five.

Once the robot part has been made the next stage is construction of a base. That is not critical and therefore detailed instruction has been omitted. Briefly, however, it can be made from wood or from a plastic or metal projects box. A diagram of minimum sizes is shown in figure six.

A suitable hole is cut in the top of the box to accommodate the servo. Once the hole has been cut, the servo is screwed into place. The platform

MATERIALS

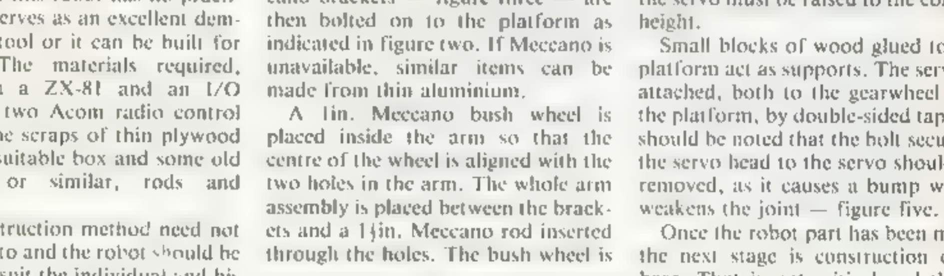
The servos — Acom AS-1 — used cost between £12 and £15. They may be obtained from any model shop which supplies Acom radio control equipment.

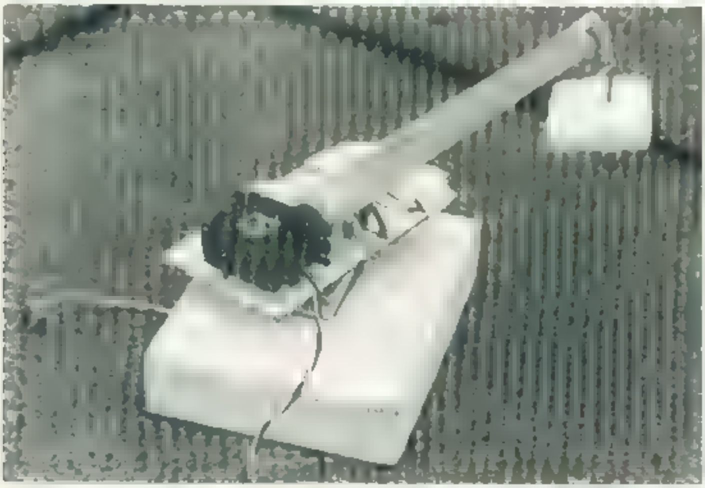
Since the number of stockists is enormous, the best way of finding a supplier of the servos would be to consult advertisements in an appropriate magazine.

The I/O board was obtained from Technomatic Ltd and cost £11.50 plus VAT.

Figure 1. All sizes in millimetres.

Materials — thick card or 1/32in. ply.





can then be attached to the base. To do that a servo head is either taped — double-sided — or glued to the underside of the platform — figure two.

The platform should then be balanced. The robot arm is fully outstretched and small weights are taped on to the platform in appropriate positions so that the platform is balanced about the centre of the servo head. The servo head attached to the platform is then connected to the servo on the base.

The final step involves constructing a hook for the end of the robot arm. That is made from a paper clip bent into the correct shape. The completed hook is glued to the end of the arm. The robot is then complete and it is ready to be connected to the computer.

So that the robot may be controlled, it must be linked to the computer via an interface board. In the prototype a Technomatic I/O board was used. It consists of eight inputs and eight outputs but only two outputs are required. Other output boards may be used but the machine code routine — figure nine — must be altered as in figure nine. The servos are connected to each line as shown in figure seven. Once that is complete the robot is ready to be programmed.

The radio control servos are controlled by changing the input pulse width between 1,000µS and 2,000µS. For simplicity there are 100 programmable positions, i.e., for every 10µS the input pulse is increased the servo will turn about one degree. The time between each pulse is of the order of 18,000µS. Figure eight illustrates the waveform required. The servos in the robot are controlled by a Z-80 machine code routine which is controlled by a Basic program. The machine code routine can be broken into three parts — the 18,000µS delay, the 1,000µS pulse, and the adjustable part of the pulse, lasting 0-1,000µS. The machine code routine and accompanying comments are in figure nine.

Unless you have an assembler the machine code will have to be loaded into the computer in decimal, using the loader program — program one. To enter the code the loader program is entered into the computer. The program is RUN and the decimal values — under the column Decimal — from figure nine are entered one at a time, pressing NEWLINE after each value.

Once all the values have been entered, press STOP then NEWLINE. That will terminate the loader program which is then deleted except for the REM statement in line one which will have taken ■ different appearance. Great care must be taken when entering the machine code, as one mistake can crash the program. If that happens the plug must be removed and the loading procedure repeated.

Figure 3. Meccano part 12B (2 required).

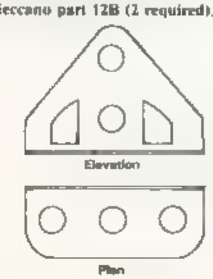
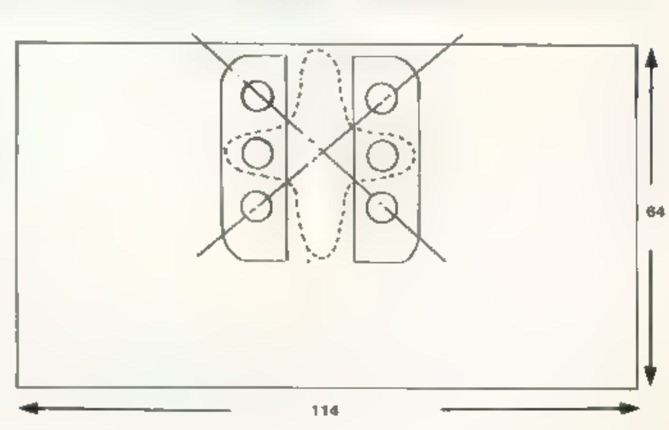


Figure 2. Pattern for platform showing position of the Meccano brackets and the servo head.



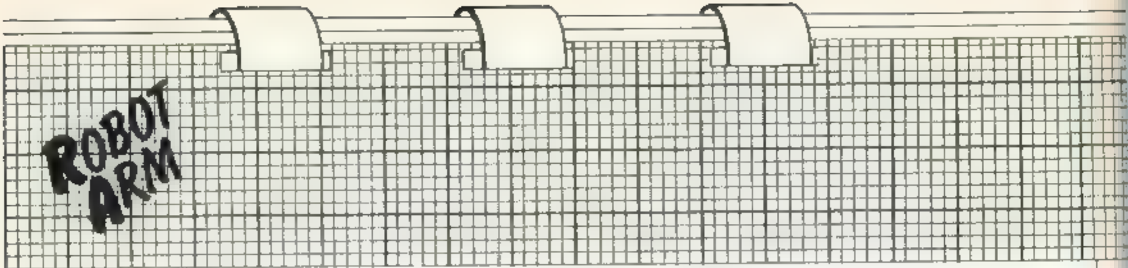


Figure 4. Not to scale or proportion.

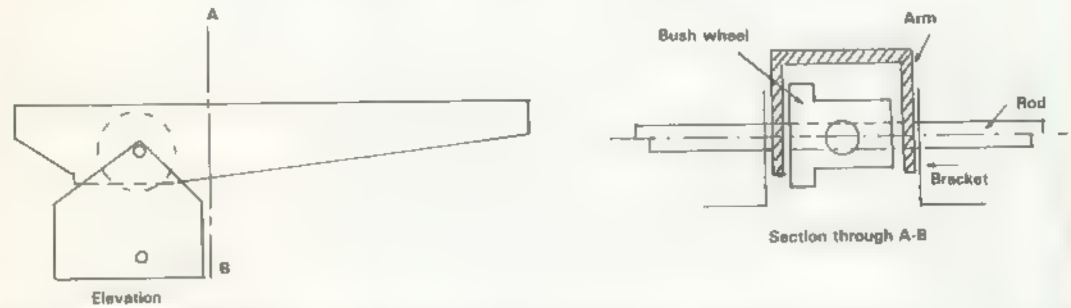


Figure 5. Not to scale or proportion.

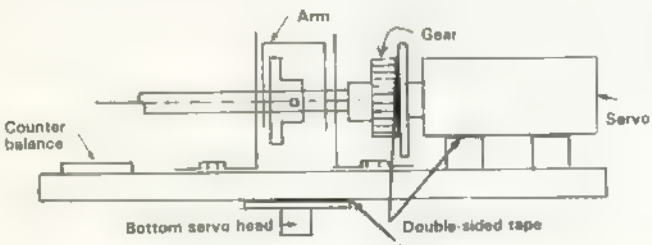
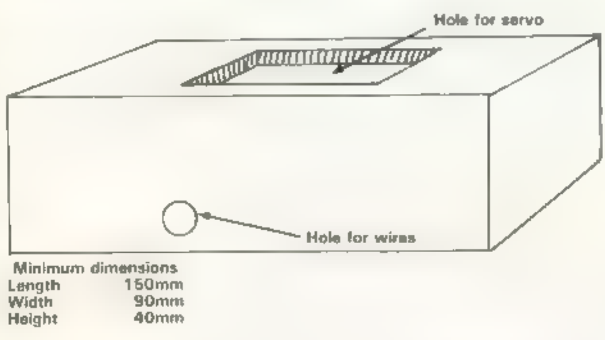


Figure 6. Not to scale or proportion.



Program 1: Machine code loader

```
1 REM [illegible]
20 FOR F = 16515 TO 40000
30 PRINT F
40 INPUT E
50 PRINT E
60 POKE F,E
70 SCROLL
80 NEXT F
```

The Basic programs controlling the machine code can then be entered. There are two such programs. The first — program two — is for manual control. In it the position of the servo motor is entered and the computer responds immediately. Program three is for automatic control. In this case the servo positions are stored in a Basic array. The program is used where repetitive movement is required — perhaps to demonstrate a production line robotic arm.

To operate the first, lines 20 to 1000 are entered. Lines 240 and 250 may be altered to suit the design of the robot. They have been included in case the robot has a limit to its movement, in which case the servo could pull the robot apart.

When the program is RUN a small menu is displayed. It indicates what code has to be entered for a certain movement. A number is entered which corresponds to a command, e.g., entering 1 means that any movement will be in the vertical axis — the arm will move. The computer then displays a prompt for the absolute position. An integer between 1 and 99 is entered and the computer will move immediately to the required position. Entering 50 centres the servo.

The second program operates by poking the required data into the machine code routine from a Basic array. Each movement requires two pieces of data — the axis of movement and the absolute position. The first piece of information is the axis of movement — 1 = horizontal (turntable); 2 = vertical (arm). The absolute position is an integer between 1 and 99.

Program 2: Manual control.

```
20 FAST
30 PRINT " ENTER NUMBER
CORRESPONDING TO REQUIRED
MOVEMENT. 1 = VERTICAL
3 = HORIZONTAL."
40 INPUT I
50 PRINT I
80 IF I = 3 THEN POKE 16536,1
90 IF I = 1 THEN POKE 16536,2
220 PRINT "ENTER ABSOLUTE
POSITION."
230 INPUT Q
240 IF I = 3 AND Q>90 OR Q<1 THEN
GOTO 220
250 IF I = 1 and Q>75 OR Q<5 THEN
GOTO 220
260 POKE 16518,Q
270 LET L = USR 16515
1000 GOTO 20
```

Lines 30 and 50 have to be changed slightly if additional steps are required. For every one movement, the array is increased in size by two. Hence the DIM statement should also be increased by two. Line 40 should also be altered similarly. It should read 40 FOR F = 1 TO X STEP 2;X is increased by two for every extra movement.

The situation becomes clearer if the program is studied. In the demonstration program the data in the array should, when RUN, make the robot arm move almost fully up; rotate right; lower the arm; pause; raise the arm; rotate left; lower the arm and pause. That sequence will continue until the robot is stopped by halting the program.

On the ZX-81 all the programs have to be RUN in FAST mode. The reason is that when the computer is in SLOW mode the program is interrupted many times each second so that the display may be updated. In that case the critical timing required may be upset.

The servo routine and programs are not limited to the robot described. The same servos and software may also be used in other projects where critical and controlled movement is required.

The robot will not weld cars but, in addition to being an interesting toy, should demonstrate what industrial robots do.

Figure 9: Machine code listing.

Mnemonic	Address (ZX-81)	Decimal	Comments
LD C,50	16515	14	Load C with 50 for 50 pulses.
	16516	50	
LD E,50	15617	30	Load E with value for servo
	16518	50	position: 1 ≤ E ≤ 99.
LD HL,3144	16519	33	Load HL with 3144 for about
	15620	72	18 000μS delay
	16521	12	
LD A,0	16522	62	Load A with 0 for compare
	16523	0	purposes.
NOP	16524	0	
DEC HL	16525	43	
CPH	16526	188	
JRNZ	16527	32	Jump to address 16525 if
	16528	252	H ≠ A.
CPL	16529	189	
JRNZ	16530	32	Jump to address 16525 if
	16531	249	L ≠ A.
LD HL,**	16532	33	Load HL with output port address.
	16533	248	Change values if another output port
	16534	42	having another address is used.
LD (HL),1	16535	54	Set bit 0 of port high (this
	16536	1	can be changed depending on
			the servo being used).
DEC E	16537	29	
CPE	16538	187	
NOP	16539	0	
NOP	16540	0	
NOP	16541	0	
NOP	16542	0	
NOP	16543	0	
JRNZ	16544	32	Jump to address 16537 if
	16545	247	E ≠ A.
NOP	16546	0	
LD B,181	16547	6	Load B for delay C (Fig. 3).
	16548	181	
NOP	16549	0	
NOP	16550	0	
DEC B	16551	5	
CPB	16552	184	
JRNZ	16553	32	Jump to address 16551 if
	16554	252	B ≠ A.
LD (HL),0	16555	54	Set output port low
	16556	0	
DEC C	16557	13	
CPC	16558	185	
JRNZ	16559	32	Jump to address 16517 if
	16560	212	C ≠ A.
RET	16561	201	Return to Basic.

** 11000 for Technomatic I/O Port — 248 42
36850 for Latch Card — 242 143

Program 3: Automatic control.

```
20 FAST
30 DIM L(16)
40 GOSUB 1000
50 FOR F = 1 TO 16 STEP 2
60 POKE 16536,L(F)
70 POKE 16518,L(F + 1)
90 LET L = USR 16515
100 NEXT F
110 GOTO 50
1000 LET L(1) = 1
1002 LET L(2) = 90
1004 LET L(3) = 2
1005 LET L(4) = 1
1007 LET L(5) = 1
1008 LET L(6) = 10
1010 LET L(7) = 1
1011 LET L(8) = 10
1013 LET L(9) = 1
1014 LET L(10) = 50
1016 LET L(11) = 2
1017 LET L(12) = 90
1019 LET L(13) = 1
1020 LET L(14) = 10
1022 LET L(15) = 1
1025 LET L(16) = 10
6000 RETURN
```

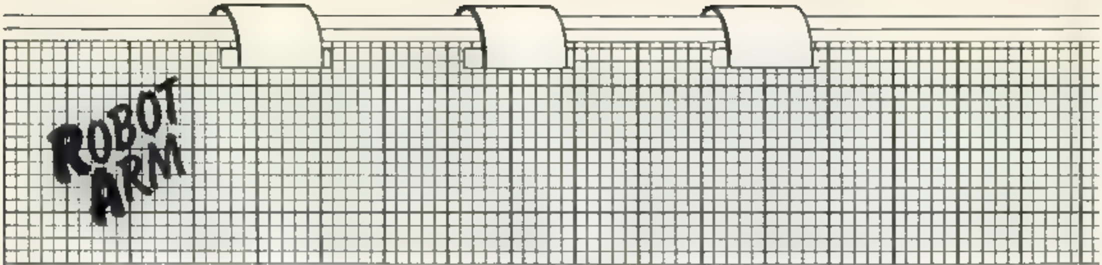


Figure 7.

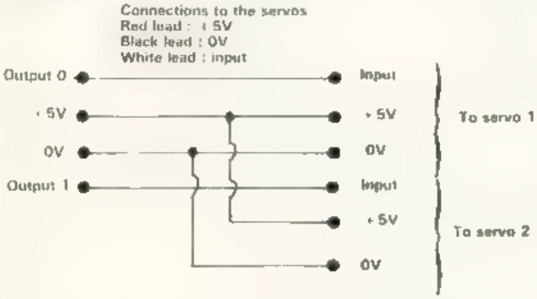


Figure 8.

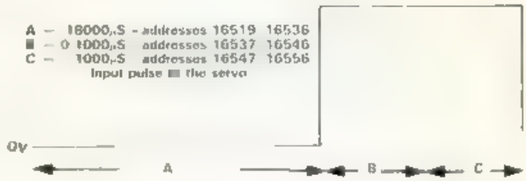
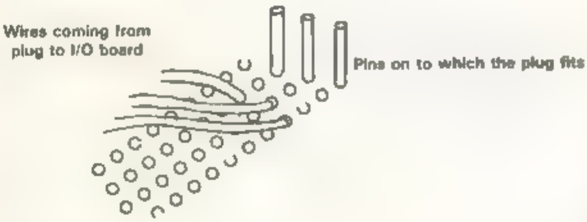
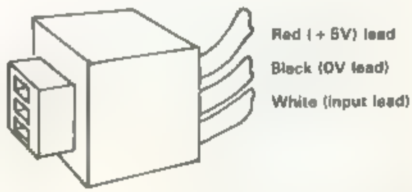


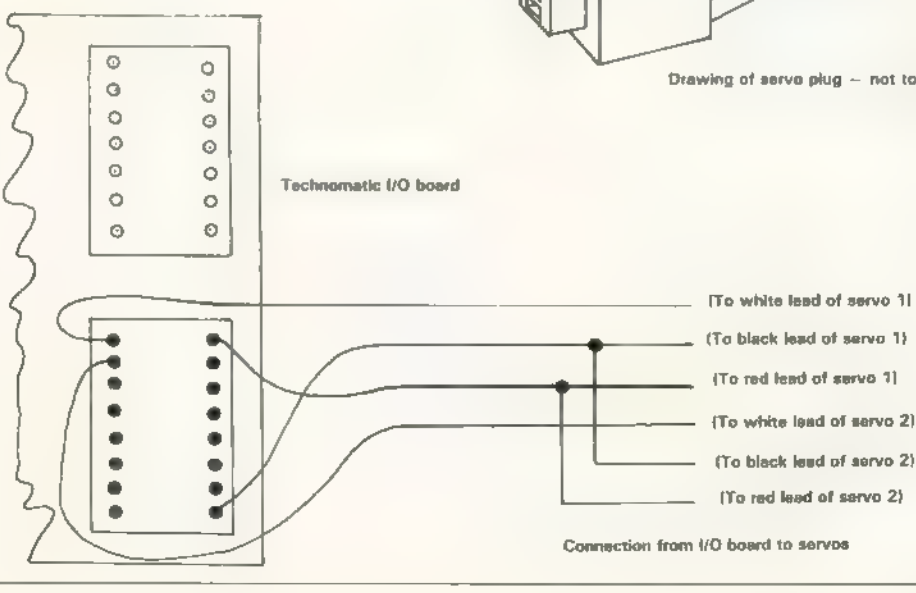
Figure 10.

The plugs were connected to the wires using Veropins stuck in an experimentor board. Veroboard could also be used if the pins were soldered in. Pins were used because the spacing matched the plug.





Drawing of servo plug — not to scale



Connection from I/O board to servos


DIGITAL ELECTRONICS

Reducing power demand by mixing TTL and CMOS

Joe Pritchard continues his series on electronic theory, looking at interfacing different types of devices, buffers and other interesting areas

THIS TIME we will look at the interfacing of TTL and CMOS devices, at buffers, three-state devices and the computer databus. Why is it necessary to mix TTL and CMOS devices in circuitry? There are several reasons. One of them is that the CMOS devices load whatever they are connected to by only a small amount — less than that for an LSTTL device. They also consume less power. Also there are some functions available in the CMOS family which are not available in the TTL family.

CMOS units can also drive more CMOS inputs than can LSTTL units drive LSTTL inputs. CMOS devices will function on 5 volts, like TTL, but they will work satisfactorily up to 15 volts. Thus we might make an address decoder circuit from CMOS devices to minimise the loading effect on the computer bus and then interface it to LSTTL devices for the rest of the circuit.

Table one shows the input and output voltage characteristics of CMOS and LSTTL devices. They are typical values for the devices and so in practice some variation from the figures is to be expected. Let us consider the case of interfacing a CMOS device to an LSTTL input. The following conditions must be met:

i V_{OH} for CMOS must exceed V_{IH} for TTL.

ii V_{OL} for CMOS must be less than V_{IL} for TTL

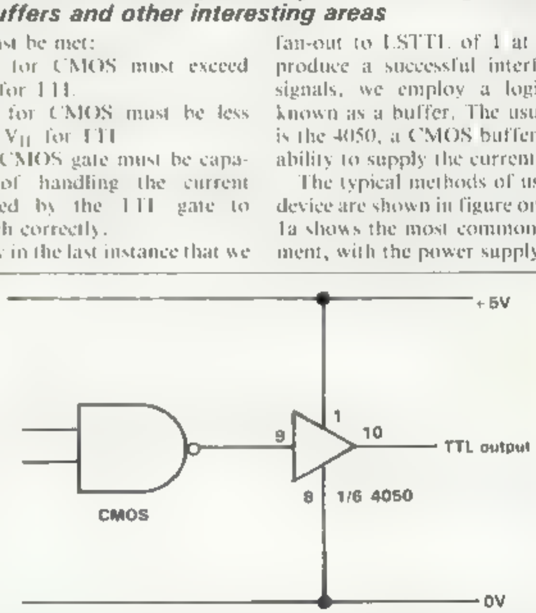
iii The CMOS gate must be capable of handling the current needed by the TTL gate to switch correctly.

It is only in the last instance that we need to take care. The TTL input needs about 0.3mA when a low input is applied and the CMOS output is not capable of providing that current. The device may be able to provide the current but it is not guaranteed and so we do not want to rely on it.

The CMOS gate has, therefore, a fan-out to LSTTL of 1 at best. To produce a successful interfacing of signals, we employ a logic device known as a buffer. The usual device is the 4050, a CMOS buffer with the ability to supply the current needed.

The typical methods of use for the device are shown in figure one. Figure 1a shows the most common arrangement, with the power supply voltages for the TTL and CMOS parts of the circuit being at the same level.

Figure 1a.



That need not be the case and in situations where the CMOS supply is higher than the TTL supply, the circuit configuration of figure 1b is used.

In most logic devices, applying a 1 signal with a value in volts of more than the supply rail will damage the device. That is not so with the 4050. In that application, we say that the 4050 is involved in a level conversion role — i.e., translating a voltage representing a logic 1 in one system into a voltage representing a 1 in another system. When used as a CMOS-to-TTL converter, the buffer can provide sufficient current for two LSTTL gates. The pin-out for the device is shown in figure two.

In the reverse situation, where we wish to interface a TTL output to a

Figure 1b. Using 4050 as a level changer.

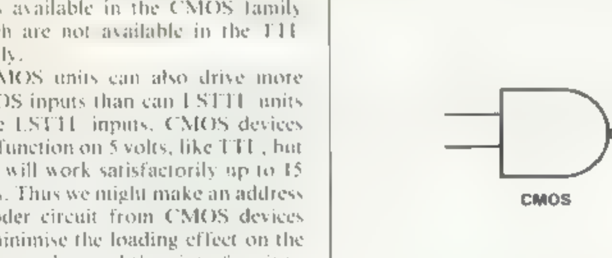


Table 1. Characteristics at 5V power supply.

Name	Description	CMOS (typical)	TTL (typical)
V_{OH}	Output high voltage	5V	3.4V
V_{OL}	Output low voltage	0.01V	0.5V
V_{IH}	Voltage for high input	3.5V	2V
V_{IL}	Voltage for low input	1.5V	0.8V

CMOS input, the situation is easier. We have no current considerations, due to the very low requirements of the CMOS devices. So all we have to consider are the voltage levels. When high, an LSTTL output is guaranteed to deliver about 2.4 volts. The minimum voltage a CMOS gate will recognise as a high input is between 3 and 4 volts.

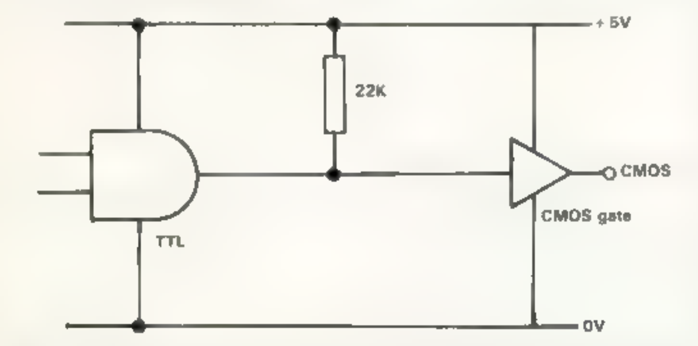
They are the worst possible cases but we must design our circuits with them in mind. The answer is to use a simple resistor as a pull-up component. Its function is to ensure that whenever the TTL output goes high, the CMOS input always sees a minimum of 3.5 volts. That is shown in figure three. Obviously the resistor should be of a value so that when the TTL output is low, the CMOS input is low as well.

We have already seen the use of a buffer device in increasing the current availability from a logic device. Also we used a 7404 device in part one of the series to help us drive a LED. We could easily have used a buffer. Buffers can be obtained in which, as well as having the ability to buffer a circuit, they also perform a logical invert function. They are called inverting buffers.

Some typical buffers are the 74LS16 and the 74LS17 devices. The 16 device is a hex inverting buffer and the 17 is a non-inverting device. The 4049 is a CMOS hex inverting buffer. As well as providing an interfacing function, we can use a TTL buffer to increase the fan-out of other LSTTL devices, e.g., feeding a standard TTL buffer from an LSTTL output and then taking the buffer output to other LSTTL devices.

A final use of buffers is in the field of protecting delicate chips from human beings. If we are using an advanced chip such as a parallel input/output device, known commonly as a PIO, then in experimental work or in education it is a good idea to input signals to the PIO only through buffers. In that way, if a high voltage was applied accidentally the buffers would be damaged and not the more expensive PIO.

Figure 3. TTL-to-CMOS interface.



All the buffers we have mentioned so far and which have belonged to the TTL family have had what are known as totem pole outputs. That rather graphic description refers to the internal design of the chips. There are some buffers which have a different type of internal circuitry at their outputs, while still being members of the TTL family.

Those devices are said to have open collector outputs. The main difference so far as we are concerned is the current and voltage ratings of the two output types. The open collector outputs are capable of handling more current, typically up to 40mA. The 7407 and the 7406 are open collector driver devices; the phrase driver is used rather than a buffer because those devices can handle more current. Applications are shown in figure four.

Table 2. Control of LS245.

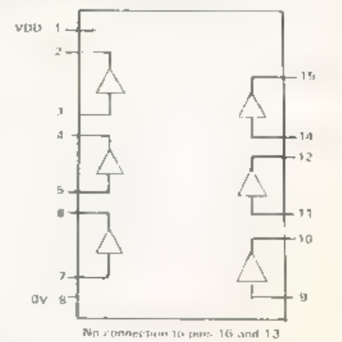
$\overline{CE}$	DIR	Operation
0	0	0 + 0I
0	1	I to 0
1	0	float
1	1	float

The 7406 device has an inverting function and the 7407 a non-inverting function. In both cases, current flows through the load and into the TTL output when the output is low.

It is not only buffers which are available with the open collector outputs; other logic devices have them as well. A typical example is the 7401, which is the open collector equivalent of the 7400 device. An interesting side-effect of the open collector device is that it enables you to construct some logic functions without using logic gates. An example is the wired AND gate shown in figure five and its practical form in figure six. The totem pole output devices make this practice unwise but it is easily and safely implemented on the open collector devices.

Readers wanting an explanation

Figure 2. Pin-out for 4050.



Digital Electronics

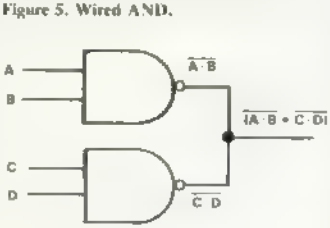
are advised to seek a reference book on the subject, as to explain it here would involve a rather detailed examination of the internal design of the TTL family of devices. Suffice to say that the 'wired AND' function is just one of a series of wired logic functions which can be formed in this way from open collector devices. That is obviously important in the design of commercial logic devices but is not so relevant for the home designer, for whom the extra expense is usually not too great.

We now discuss a group of devices which are extremely important in the design of devices to fit on to the computer databus. They are the three-state buffers, also known as tri-state buffers. A three-state device has three output states. I know that one of the first things we learned about logic devices was that there are two logic states, 1 and 0. Well, the three-state devices have those two states and a third state, known coloquially as the "float" state.

In that condition, the output is said to present a high impedance to any device connected to it. When floating, the output has no effect on a device connected to it. It is as if we had a switch by which we could disconnect the output of the three-state device from the input of the next device. When the output is floating, the three-state device is said to be disabled. When we allow the output to assume a 1 or 0 logic value, we say that the device is enabled.

Control over whether a device is enabled or disabled is done via a pin on the device. Figure seven shows the circuit symbol for a non-inverting

Figure 5. Wired AND.



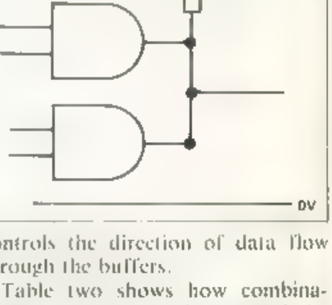
three-state buffer. Let us examine a few practical three-state buffers in the TTL family. The simplest is probably the 74LS125, which is a quad tri-state buffer. The pin-out for the device is shown in figure eight. The circle on the enable line indicates that it is an active low line, i.e., the line is taken to a logic 0 to enable the gates. The 74LS126 is identical except for the small difference that the enable line is taken high to enable the gates.

In each of those devices, there is an individual enable line for each gate in the package. That is not usually the case, as often there are up to eight gates in the package. So the enable lines are said to be commoned, in that several gates are controlled by one enable line. An example is the 74LS244 device which has eight buffers controlled as two blocks of four gates. Figure nine shows an application for this device. When the control signal is low, the data on the eight lines from the external logic is made available for the databus of the computer. The control signal could be generated by the computer. When the signal goes high again, it is as if the LS244 were not connected to the databus of the computer.

All the buffer chips considered have been uni-directional — in a certain circuit the device has to be re-wired to allow a signal to go in the other direction. It would be very useful if devices existed which would allow signals to flow in both directions depending on the state of a control signal to the device.

Such a chip exists, the LS245. It is a TTL device and allows two-way or bi-directional data flow depending on the state of one of its pins. The pin-out for the device is shown in figure 10. The $\overline{CE}$ terminal of the chip is the enable line. That is an active low line, as indicated by the bar over the label, and is taken low to enable the buffers. The DIR pin is the pin which

Figure 6. Open collector output wired AND.



controls the direction of data flow through the buffers.

Table two shows how combinations of those two pins control the buffers. Here, all eight buffers are controlled by the lines simultaneously. Obviously, the device will allow two-way communication between the data lines connected to it. If we enabled the chip and then took DIR to a logic low, data would flow from the B

Figure 4a. Inverting buffer.

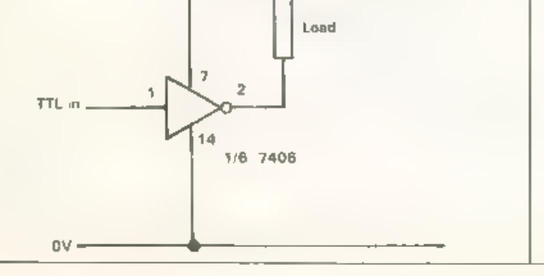


Figure 4b. Non inverting buffer.

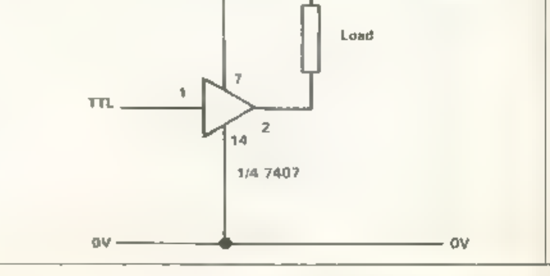
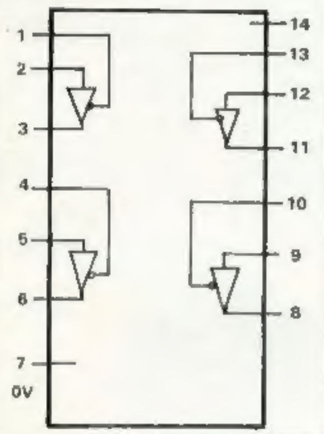


Figure 8. LS125 pin-out.



lines to the A lines. The lines connected to B are said to be transmitting data and the A lines are said to be receiving data. If we took the DIR line high, the situation would be reversed. As eight lines are involved, the device is often described as an octal bus transceiver.

Other three-state devices include the LS367 and the 81LS97. The latter unit is a member of a family of devices designed originally for use with the 8080 microprocessor but is useful in other applications. None of the devices mentioned so far has been latched. The idea will be discussed in detail next time but the basic concepts are that the devices so far examined have, when enabled, re-created at their outputs the data which was at their inputs at that moment.

A latched device can remember data which has been put there by a circuit until it is read. Even after it has been read, the latch can still hold the data until it is told to forget the data.

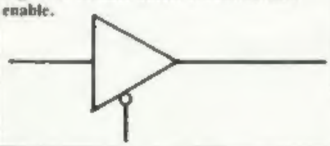
The last operation is called re-setting the latch. Thus the latched buffers are handy devices. In a previous article, in the June-July issue of *Sinclair Projects*, I discussed the 8212 latch in some detail. As we have mentioned the computer bus in passing this time, let us look at it in more detail.

In a computer, the various components which make up the device are linked by wires or tracks on printed circuit boards. Some lines, as the wires are called, carry information about the present state of the computer and they are called control signals, as they control interactions between the microprocessor at the heart of the computer and the other devices in the circuit.

Those lines constitute the control bus. Other lines carry information relating to what the computer should do next and on what numbers the computer should next operate. The lines are the databus.

A further set of lines tells the computer where it can find its next instruction or data item. That is the address bus of the computer. The address bus and control bus are usually uni-directional, although some control lines will be bi-directional. The data bus is bi-directional as it must carry data to and from the microprocessor.

Figure 7. Tristate buffer with active low enable.



All of those lines carry logic signals and have a fan-out like any other logic device. With regard to the address and control buses, buffering is not usually needed so long as only a few extra LSTTL or CMOS loads are placed on the address lines.

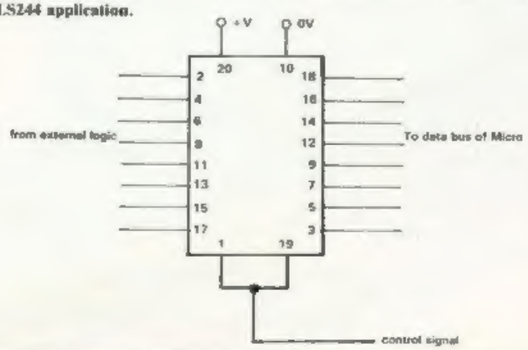
With several devices, buffering becomes necessary and could be done with some of the devices already seen. Buffering between devices added and the databus should be considered to be essential, so that when the device is not in use it does not interfere with the databus in any way. You can imagine the confusion which would confront the microprocessor if two unbuffered devices tried to send signals to it at the same time, the situation which would occur if the devices were not isolated from the bus by the three-state buffers. Circuits which send data to the databus or devices which receive signals from the bus may be isolated by the use of uni-directional three-state buffers but devices which are expected to send and receive information would need bi-directional devices.

Memory chips, such as those which make up the RAM of Sinclair micros, often have three-state buffers built into them so that they can interface directly to the databus.

The concept of the bus will be explored further in a future part of the series. Next time, we will examine the second major group of logic devices — sequential logic circuits. They are the circuits whose logical behaviour depends to a certain extent on signals applied previously.

With regard to practical experiments, this time a few words on using CMOS devices might be in order. Due to their construction, CMOS devices need protecting from static electricity, which can damage them seriously. Although protection is often built-in, a little care will make life easier

Figure 9. LS244 application.



DIGITAL ELECTRONICS

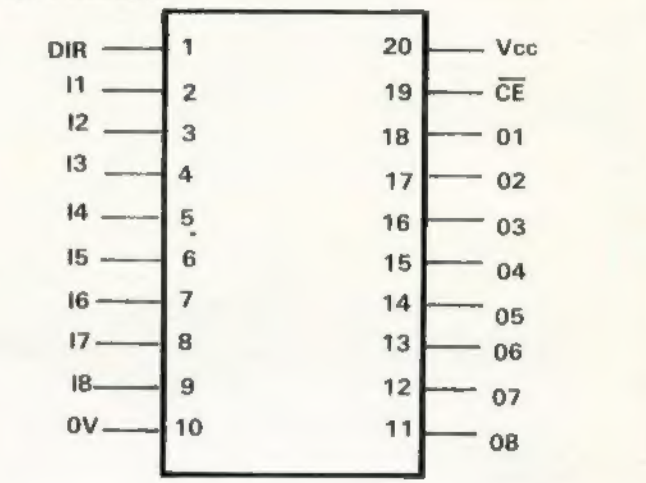
for both you and the chip. The devices are usually protected up to about 4,000 volts and a static discharge of below that should not bother them. As you can generally develop a voltage on your fingers of up to 10,000 volts by walking across a nylon carpet, care is still needed. So here are a few points to note when using CMOS devices:

- Never solder to them directly — always use sockets. That also makes fault-finding easier.
- Most CMOS devices are delivered to the user with their pins either short-circuited with conductive foam or with the pins stuck into metal foil. Never remove the chip from that protection until you are ready to use it in the circuit.
- When breadboarding circuits, insert the chip last. Never insert or remove a chip from a powered-up circuit.
- Try not to touch the legs of a CMOS device. When handling, try not to wear nylon clothing.
- Before handling, try to discharge any charge on you by touching a cold water tap or pipe.

They may seem extreme precautions but they are the ones I employ and I have not lost a CMOS chip yet.

The construction method of the CMOS family makes their inputs very susceptible to electrical noise if they are left unconnected. With TTL devices, unconnected inputs assume a logical value of 1. That is not the case with CMOS gates and it is necessary to tie all unused inputs to either logic 1 or logic 0, depending on the logic function you are trying to achieve in the circuit.

Figure 10. Pin-out of LS245.



Project buyers' guide

HERE IS a list of suppliers for difficult-to-obtain items which have been used in projects.

Edge connectors 23-way for ZX-81 and
28-way for Spectrum.
Innovonics

PCB mounting 3.5mm. jack sockets as used in the Central Heating Controller project.
MS Components Ltd

Extender cards for fitting to rear of edge connector to allow stacking add-ons.
23-way for ZX-81 – ZXTONGUE
28-way for Spectrum – SPECTONGUE
Innovonics

MS Components Ltd, Zephyr House, Waring Street, West Norwood, London SE27. Tel: 01-670 4466.

Innovonics, 147 Upland Road, East Dulwich, London SE22.

UPDATE

Errors

December 1983/January 1984

Update, page 14. Waveforms: "lower Q should be $\overline{Q}$".

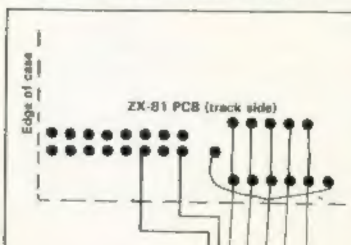
Sound Generator, page 17, figure 3: "C1 should be 330pf".

Digital logic, page 24, figure 20: "The NOT gate should be in the other input to AND gate E".

Prowler, page 27, figure 1: "The connection from R4 to IC2 goes to pin 1, $\overline{MR}$"; page 30, figure 7: "pin 15 is R/$\overline{W}$".

Battery support, page 43 circuit layout: "Connection A is by the top end of R3".

Readers' Tips — Four-button keypad, p13. The right-hand side of the figure showing the connections to the ZX-81 PCB should be as below:



Second Edition Just Published

If you only buy ONE computing book

THIS IS THE CLASSIC!

An entertaining, but **thorough** reference source with the most readable, comprehensive glossary you'll find anywhere. The Good Computing Book for Beginners is an essential A-Z of all the facts you need to know about computing — and none you don't!

Whether you're brand new to computing, or an old hand keen to stay ahead, you'll score by keeping this classic, top-selling book within reach to use again and again.

The author, Dennis Jarrett, is a successful journalist who was also founder editor of Which Computer? magazine — so his first edition quickly became a standard work. Now this new, substantially enlarged and revised edition covers the latest trends, terms and technology with the relevant facts — and ONLY the relevant facts — in **plain English.**

Here's another fact you'll find fascinating: it will only cost you £2.95!

Use the coupon below to get your copy right away — or buy it soon from your bookshop. It's the **one** book you'll turn to again and again.

Please send me ______ copy(ies) of The Good Computing Book for Beginners by Dennis Jarrett at £2.95 each plus 50p post and packing. I enclose a cheque for ______ Please debit my credit card account Visa/Access/Diners/Amex number ______

Signature ______

Name ______

Address ______ (please print clearly)

Send to: ECC PUBLICATIONS LIMITED
196–200 Balls Pond Road
London N1 4AQ

A New Dimension in Spectrum Hardware

Joystick and Interface

for Sinclair Spectrum with these features to give you endless hours of enjoyment.

1. Super positive response fire button.
2. Firm suction cups for stable one hand operation.
3. Snug fit hand moulded grip.
4. Additional fire button.
5. Extra long 4 ft lead.

The interface supplied with the Quick Shot (TM) has a two joysticks facility.

The first port simulates 6789 & 0 keys. The second port simulates in (31) command. It will run any Software.

1. Using keys 6, 7, 8, 9 and 0.
2. Having redifinable key function.
3. Using in (31) i.e. Kempston.
4. Any Software you write yourself.

£22.95

Light pen

The LIGHT PEN enables you to produce high resolution drawings on your own TV screen simply by plugging into the ear socket of your Spectrum. The controlling software supplied with the light pen has 16 pre-defined instructions. You can change colour (Border, Paper, Ink), draw circles, arcs, boxes, lines and insert text onto the screen at any chosen place, you can also draw freehand. There is a feature to retain the screens and animate. On the 48K Spectrum you can retain 5 screens. You can also use the machine code on its own in your own programs, for selecting out of a menu etc. The LIGHT PEN is supplied with a control interface, to adjust the sensitivity/pen alignment.

£19.95

Keyboard for use with a Spectrum or ZX81

£45.00

Our cased keyboard has 52 keys, 12 of which form a numeric pad. The 12 keys comprise 1-9 numeric plus full stop and shift keys, all in red, to distinguish from the main keyboard keys which are in grey, the keys contrast with the black case to form a very attractive unit. The case has been designed to take a ZX81 or Spectrum computer. 16K, 32K or 64K can also be fitted to the motherboard inside the case (81 model only). The case is also large enough for other add-ons like the power supply to be fitted, giving a very smart self-contained unit with which other add-ons e.g. printer etc. can still be used. Our ZX Professional keyboard offers more keys and features than any other model in its price range making it the best value keyboard available.

Spectra-Sound

The so-called speaker in your Spectrum is really on a 'buzzer'. With the DK Tronics "SPECTRA SOUND" you can generate fully amplified sound through the speaker on your TV set. SPECTRA SOUND is a very simple but highly effective add-on. This means that you no longer have a faint beep but a highly amplified sound, which can be adjusted with the TV volume control. The SPECTRA SOUND fits compactly and neatly inside the Spectrum case and is connected by three small crocodile clips.

£9.95

Please send me .... @ £ ....

Please send me .... @ £ ....

Please add on £1.25 for post and packing.

I enclose cheque/PO payable to DK Tronics total £ ....
or debit my Access/Barclaycard No.

☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Signature ....

Name ....

Address ....

Send to: DK Tronics Ltd., Unit 6, Shire Hill Ind. Est., Saffron Walden, Essex CB11 3AQ. Tel: (0799) 26350 (24 hrs) 5 lines

dk'tronics

DK Tronics Ltd., Unit 6, Shire Hill Industrial Estate, Saffron Walden, Essex CB11 3AQ.
Telephone: (0799) 26350 (24 hrs) 5 lines